Artificial Intelligence 195 (2013) 496–527

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Bagging and Boosting statistical machine translation systems ☆

Tong Xiao [a], Jingbo Zhu [a], Tongran Liu [b,*]

[a] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[b] Key Laboratory of Behavioral Science, Institute of Psychology, Chinese Academy of Sciences, 10A Datun Road, Chaoyang District, Beijing 100101, China

ARTICLE INFO

Article history:
Received 15 March 2011
Received in revised form 5 October 2012
Accepted 15 November 2012
Available online 12 December 2012

Keywords:
Statistical machine translation
Ensemble learning
System combination

ABSTRACT

In this article we address the issue of generating diversified translation systems from a single Statistical Machine Translation (SMT) engine for system combination. Unlike traditional approaches, we do not resort to multiple structurally different SMT systems, but instead directly learn a strong SMT system from a single translation engine in a principled way. Our approach is based on *Bagging* and *Boosting* which are two instances of the general framework of ensemble learning. The basic idea is that we first generate an ensemble of *weak* translation systems using a base learning algorithm, and then learn a *strong* translation system from the ensemble. One of the advantages of our approach is that it can work with any of current SMT systems and make them stronger almost "for free". Beyond this, most system combination methods are directly applicable to the proposed framework for generating the final translation system from the ensemble of weak systems. We evaluate our approach on Chinese–English translation in three state-of-the-art SMT systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based system. Experimental results on the NIST MT evaluation corpora show that our approach leads to significant improvements in translation accuracy over the baselines. More interestingly, it is observed that our approach is able to improve the existing system combination systems. The biggest improvements are obtained by generating weak systems using Bagging/Boosting, and learning the strong system using a state-of-the-art system combination method.

1. Introduction

Machine Translation (MT) is one of the oldest sub-fields in Natural Language Processing (NLP) and Artificial Intelligence (AI). During the last decade, statistical approaches have received a growing interest in machine translation, showing state-of-the-art performance for many language pairs. To date, several Statistical Machine Translation (SMT) paradigms have been developed, including phrase-based SMT [58], hierarchical phrase-based SMT [14], and syntax-based SMT [20,23,32,39,66]. Although these systems and approaches are of competitive translation quality, they have different strengths and weaknesses. For example, phrase-based approaches are very powerful in local reordering that is inherent in phrase translations, but have limited capabilities in dealing with long distance phrase dependencies. On the other hand, syntax-based approaches characterize the movement of hierarchical structures by linguistic notions of syntax, but suffer from the errors of syntactic parsers and structure divergence between languages. By offsetting weaknesses with strengths of other systems, combination is a

☆ This article is an extended version of a conference paper (Xiao et al., 2010 [109]). Substantial differences are: we present a new method (Bagging) for improving single SMT engines (Section 3); we present significant extensions of the experimental studies on the proposed methods (Section 4); and we present more detailed analysis of the related work (Section 5) and discussions on several important issues (Section 6).

* Corresponding author. Tel.: +86 10 64854533.
E-mail address: liutr@psych.ac.cn (T. Liu).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.11.005

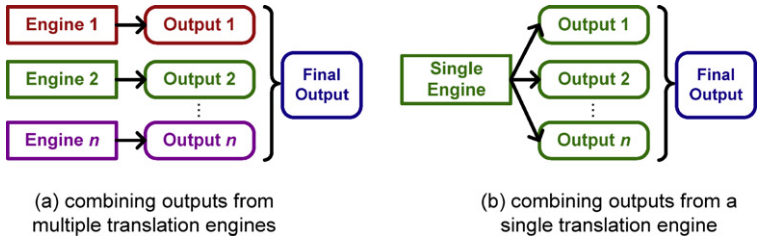

Fig. 1. Illustration of different ways to generate diversified translation outputs for system combination.

desirable way to achieve higher translation accuracy than does any individual system. Therefore, it is beneficial to explore approaches that combine different systems as well as their outputs and produce better translations. Recently, motivated by the GALE program,[1] multiple SMT systems have been developed by different consortiums, and more and more studies are focused on combining multiple SMT systems to further advance the final consortium system.

The basic idea of system combination is to extract or generate a translation by voting from an ensemble of translation outputs. Depending on how the translation is combined and what voting strategy is adopted, several methods can be used for system combination. For example, sentence-level combination [49] simply selects one from original translations, while more sophisticated methods, such as word-level/phrase-level combination [76,88], can generate new translations differing from any of the original translations.

One of the key factors in system combination is the diversity in the ensemble of translation outputs [69]. To obtain diversified translation outputs, most combination systems require multiple translation engines based on different models [6, 76,89] (see Fig. 1(a)). Although various SMT systems/models have been utilized for system combination, previous approaches only use the ensemble of existing systems at hand, rather than generating the ensemble of different systems in a principled way. Moreover, due to the high cost of developing and tuning SMT systems, these approaches might not be very good choices when multiple SMT engines cannot be accessed. In this case, it is worth exploring alternatives that combine translation systems built from a single translation engine instead (Fig. 1(b)).

In this article, we address the issue of how to generate an ensemble of diversified translation systems from a single translation engine in a principled way. Our contribution is two-fold:

- We present a simple and effective approach to learning a strong translation system from a single SMT engine using Bagging and Boosting.
- We present a comprehensive evaluation of Bagging and Boosting on SMT system combination, and demonstrate their effectiveness in improving translation quality for three types of SMT models and several existing system combination methods.

In our approach, we first generate a diverse ensemble of *weak translation systems* (or *weak systems* for short) by manipulating a distribution over the training data, and then combine them to produce a *strong translation system* (or *strong system* for short). To obtain the ensemble, a sequence of weak systems is generated from a base system in an iterative manner. In each iteration, a new weak system is learned with respect to a distribution over training samples. Two strategies are adopted to maintain the distribution: generating different versions of the training set by using sampling with replacement (as in Bagging), or changing the distribution to focus more on the training samples that are relatively poorly translated by previous weak systems (as in Boosting). After all the weak systems are collected, a *strong* translation system is finally built from the ensemble of weak translation systems using various system combination methods.

Such an approach has several benefits. First of all, as no restriction is made on what type of translation engines we learn from, it can work with any SMT systems and make them stronger almost "for free". Moreover, current system combination methods are directly applicable to generating the strong system from the ensemble of weak systems. This means that we can re-use existing system combination methods for the final system generation in the proposed framework, rather than developing new methods. In addition, our approach is very "cheap" from the engineering standpoint. All we need is a slight modification of the weight training module, while keeping other components unchanged in current SMT pipeline (if the system combination method has already been repaired for selecting/generating the final translation from the ensemble).

Our experiments are carried out on Chinese–English translation in three state-of-the-art SMT systems, including a phrase-based system [40,110], a hierarchical phrase-based system [14] and a syntax-based system [39,70]. These systems are evaluated on the NIST MT evaluation test sets. Experimental results show that our approach leads to significant improvements in translation accuracy over the baseline systems. More interestingly, it is observed that the proposed approach is very helpful in improving existing system combination methods. For example, the biggest improvements are obtained by generating weak systems using Bagging/Boosting, and learning the strong system using a state-of-the-art system combination

[1] http://www.darpa.mil/ipto/programs/gale/index.htm.

method. When combining three types of translation engines, our approach finally results in an absolute improvement of over 1.9 BLEU points over the baselines on all the evaluation sets.

The rest of the article is structured as follows. Section 2 briefly introduces the background knowledge of SMT and ensemble learning. Section 3 describes our approach to generating and combining diversified translation systems with Bagging and Boosting. Then, Section 4 presents experimental evaluation of our approach. After reviewing the related work in Section 5, some interesting issues are discussed in Section 6. Finally, the article is concluded with a summary and outlook of further work.

2. Background

2.1. Log-linear model and statistical machine translation

Given a source string f, the goal of machine translation is to translate f into a target string e. In SMT, this problem can be stated as: we find a target string e^* from all possible translations by the following equation:

$$e^* = \arg\max_e \Pr(e|f) \tag{1}$$

where $\Pr(e|f)$ is the probability that e is the translation of the given source string f. To model the posterior probability $\Pr(e|f)$, most SMT systems utilize the log-linear model proposed by Och and Ney [81]:

$$\Pr(e|f) = \frac{\exp(\sum_{m=1}^{M} \lambda_m \cdot h_m(f,e))}{\sum_{e'} \exp(\sum_{m=1}^{M} \lambda_m \cdot h_m(f,e'))} \tag{2}$$

where $\{h_m(f,e) \mid m = 1,\ldots,M\}$ is a set of *features*, and λ_m is the *feature weight* corresponding to the m-th feature. $h_m(f,e)$ can be regarded as a function that maps each pair of source string f and target string e into a real number, and λ_m can be regarded as the contribution of $h_m(f,e)$ to $\Pr(e|f)$. Ideally, λ_m indicates the pairwise correspondence between the feature $h_m(f,e)$ and the overall score $\Pr(e|f)$. A positive value of λ_m indicates a correlation between $h_m(f,e)$ and $\Pr(e|f)$, while a negative value indicates an inversion correlation.

In this article, we use u to denote a model that has M fixed features $\{h_1(f,e),\ldots,h_M(f,e)\}$, use $\boldsymbol{\lambda} = \{\lambda_1,\ldots,\lambda_M\}$ to denote the M parameters of u, and use $u(\boldsymbol{\lambda})$ to denote an SMT system based on u with parameters $\boldsymbol{\lambda}$. In a general SMT pipeline, $\boldsymbol{\lambda}$ is learned on a training (or tuning) data set[2] to obtain an optimized weight vector $\boldsymbol{\lambda}^*$ as well as an optimized system $u(\boldsymbol{\lambda}^*)$. To learn the optimized weight vector $\boldsymbol{\lambda}^*$, $\boldsymbol{\lambda}$ is usually optimized according to an objective function that (1) takes the translation quality into account; (2) and can be automatically learned from MT outputs and reference translations (or human translations). For example, we can use BLEU [84], a popular metric for evaluating translation quality, to define an error function and learn optimized feature weights using the minimum error rate training method.

In principle, the log-linear model can be regarded as an instance of the discriminative model which has been widely used in NLP tasks [8,86,101]. In contrast with modeling the problem in a generative manner [13], discriminative modeling frees us from deriving the translation probability for computational reasons and provides capabilities to handle the features that are able to distinguish between good and bad translations [68]. In fact, arbitrary features (or sub-models) can be introduced into the log-linear model, even if they are not explained to be well-formed probabilities at all. For example, we can take both phrase translation probability and phrase count (i.e., number of phrases used in a translation derivation) as features in such a model. As the log-linear model has emerged as a dominant mathematical model in SMT in recent years, we choose it as the basis of this study.

2.2. Ensemble learning

Given a model (e.g., the log-linear model mentioned above), many individual statistical systems with different parameters (e.g., different features weights in the log-linear model) can be learned, and the "best" system is in general selected according to some criteria. Though widely used, this paradigm suffers from two problems. First, we have to decide *which one is the best*. The answer is not so straightforward when more than one candidate systems meet the criteria. The situation is even more severe when the learning algorithm is prone to different local optimal and/or no sufficient training data is provided. Second, potentially-valuable information may be lost by discarding those less-successful systems. In many cases, the discarded candidate system may successfully handle some samples, though it does not obtain the best performance over the whole data-set. A natural solution to these problems is employing multiple systems and combining their predictions. *Ensemble learning* is such a family of learning approaches, which is motivated by the fact that errors can be complemented by other correct predictions [87]. In ensemble learning, an *ensemble* is viewed as a collection of *member systems* (or *component systems*), which can be combined to obtain a stronger generalization ability than the individual systems making up the ensemble. Generally, member system is also called weak system, and the ensemble of weak systems is called strong system.

[2] The data set used for weight training is generally called *development set* or *tuning set* in the SMT community. In this article, we use the term *training set* to emphasize the training of the log-linear model.

Typically, an ensemble is built in two ways. First, a number of weak systems are learned using a *base learning algorithm* (or *base learner*). Then, the resulting weak systems are combined to generate a strong system. To learn a good ensemble, a widely-adopted principle is that the weak systems in the ensemble should be both *accurate* and *diverse* [43,45,59,83]. In general, learning accurate individual system is task-oriented: there is no "best" learning algorithm for all the problems; and we need to select an appropriate learning algorithm for a specified task. Therefore, most ensemble learning approaches focus more on generating and combining a diverse and complementary set of weak systems [10,50,91,106]. To introduce diversity into the ensemble, there are many choices [3,52,71,105], such as subsampling training samples, manipulating outputs of weak systems and injecting randomness in base learning algorithm. Among them, two most popular methods for creating accurate and diverse ensemble are Bagging [10] and Boosting [91]. These methods are based on resampling techniques which obtain different distributions of training set for each of the individual systems. Previous work has demonstrated that both Bagging and Boosting are effective for classification systems [7,10,11,25,37]. However, there have been few empirical and/or theatrical studies with MT systems (especially with MT system combination).

2.3. Adapting Bagging and Boosting to SMT

We believe that Bagging and Boosting are promising in SMT for several reasons. First, ensemble learning is a well-studied topic in Machine Learning (ML). Previous studies have shown that Bagging and Boosting have very nice theoretical properties, which leads to good ability in reducing both training error and generalization error [82,92]. Moreover, empirical testing has verified the effectiveness of these methods in several NLP tasks, such as text categorization [94] and natural language parsing [48]. It also indicates the potential power of Bagging and Boosting in MT. In addition, ensemble learning methods address both *the statistical problem* and *the computational problem* which most SMT systems suffer from. As stated in [21], the statistical problem refers to the case that the learning algorithm has to select 1-best result from several equally-good systems. For example, translations generated by different SMT systems are sometimes of very competitive translation quality (e.g. comparable BLEU scores). Obviously, voting from these systems can reduce the risk of missing the "best" translation. The computational problem actually refers to the search problem in SMT. As ensemble learning can be regarded as a procedure of multi-path search in a general search framework [90], it is considered to be useful in reducing search errors. In a sense, both the statistical problem and the computational problem indicate some sort of variance for a statistical learning system. Dietterich [21] summarized them as *output variance* and *computational variance*, respectively. He further pointed out that the success of ensemble methods can be attributed to their ability in reducing the variance of learning algorithm. Many experimental studies have confirmed this argument [3,52,71,105], and imply a potential application of Bagging and Boosting in SMT.

While Bagging and Boosting have been extensively investigated in machine learning, their adaptation to SMT is not a trivial task. The first issue is how to build an ensemble of diverse SMT systems. Modern SMT systems are very complex systems which rely on several components to generate final translation. Training this type of system is not simple in either theoretical or engineering aspect. Moreover, the training criterion of SMT system is very different from that used in classification task. For example, BLEU, the well-known evaluation metric used in defining SMT training criterion, is a discontinuous and non-differentiable function, which leads to very different optimization algorithms from those used in classification. Another issue is how to combine SMT systems. In classification tasks, the search space is a predefined set of labels. By contrast, SMT systems search for the best translation in an extremely large (or almost infinite) set of target-language sentences. As a result, we have to use very different ways for combining SMT systems, which are usually much more complicated than the simple voting strategy used in classification systems.

Although a few past studies have tried to introduce Bagging and Boosting into SMT [30,62], they fail to establish a general connection between Bagging/Boosting and SMT. Some of them restrict themselves to specific SMT engines [62], and others focus only on late-stage re-ranking which suffers greatly from search errors [30]. To our knowledge, it is rare to see systematic and extensive testing of Bagging and Boosting in SMT. In this article we address all these issues. In particular, we present a general solution to adapting Bagging and Boosting to SMT (Section 3), as well as a systematic and empirical study on Bagging and Boosting SMT systems (Section 4).

3. Boosting and Bagging single translation engines

3.1. Overview

We begin by denoting the combination model as v. Then we define $v(\cdot)$ to be a *combination system* which combines a number of input member systems. The input of function $v(\cdot)$ is an arbitrary number of SMT systems, and the output is a "new" system that selects or generates the "best" translation from the k-best outputs of the input systems. By using these notations, the task of system combination can be defined as: given T SMT systems $\{u_1(\boldsymbol{\lambda}_1^*), \ldots, u_T(\boldsymbol{\lambda}_T^*)\}$, we build a new translation system $v(u_1(\boldsymbol{\lambda}_1^*), \ldots, u_T(\boldsymbol{\lambda}_T^*))$ from $\{u_1(\boldsymbol{\lambda}_1^*), \ldots, u_T(\boldsymbol{\lambda}_T^*)\}$. Here $v(u_1(\boldsymbol{\lambda}_1^*), \ldots, u_T(\boldsymbol{\lambda}_T^*))$ is the combination system which combines translations from the ensemble of the output of each individual member system $u_i(\boldsymbol{\lambda}_i^*)$. As discussed in Section 1, the diversity among outputs of member systems is an important factor contributing to the success of system combination. To obtain diversified member systems, traditional approaches concentrate more on using structurally different

Input: an SMT model u, a combination model v, a training set S consisting of m samples (denoted as $\{(f_1, \mathbf{r}_1), \ldots, (f_m, \mathbf{r}_m)\}$ where f_i is the i-th source sentence), and $\mathbf{r}_i$ is the set of reference translations for f_i.
Output: a strong translation system

1. **Initialize**: $D_1(i) = 1/m$ for all $i = 1, \ldots, m$
2. **For** $t = 1, \ldots, T$
   a) **Training**: Train a weak system $u(\boldsymbol{\lambda}_t^*)$ on $\{(f_i, \mathbf{r}_i)\}$ using distribution D_t
   b) **Re-weighting**: Generate a new distribution D_{t+1} by updating D_t
3. **Output the final (strong) system**:

$$v(u(\boldsymbol{\lambda}_1^*), \ldots, u(\boldsymbol{\lambda}_T^*))$$

Fig. 2. Overview of our approach.

member systems, that is, for any u_x and u_y in $\{u_1, u_2, \ldots, u_T\}$, we have $u_x \neq u_y$. However, this constraint condition cannot be satisfied when multiple translation engines are not available.

In this article, we argue that the diversified member systems can also be generated from a single engine $u(\boldsymbol{\lambda}^*)$ in a principled way. The first issue that arises is how to adjust the training parameters to obtain different SMT systems. In the general pipeline of SMT training [14,58], there are several key steps that affect the resulting system, for example, word alignment, phrase/rule extraction, parameter estimation, weight training (or tuning) and so on. As these training steps are mutually correlated, it is difficult to adjust them all together. Instead, a better method is to vary one training step at a time and keep others unchanged. Motivated by this idea, we focus more on the weight training (or tuning) for learning SMT systems, and obtain diversified member systems using different weight training results. We choose weight training as the base learning procedure because it has great impact on training log-linear model-based systems [79]. Moreover, since weight training is a late-stage sub-routine in the training pipeline, it could result in a more efficient method to generate diversified SMT systems, without calling early-stage sub-routines, such as word alignment. It is worth noting that, although we will restrict ourselves to weight training for our discussion and experiments, the approach presented in this article could handle a more general case where diversified member systems are generated by adjusting the parameters in early training steps.[3]

Then, assuming $u_1 = \cdots = u_T = u$, our goal can be stated as: to find a series of $\boldsymbol{\lambda}_i^*$ and build a combined system from $\{u(\boldsymbol{\lambda}_i^*)\}$. To do this, we propose an approach to generating and combining multiple translation systems using Bagging and Boosting. Like standard algorithms in ensemble learning, such as AdaBoost [37,92], this approach uses weak systems to form a strong system by repeatedly calling a base learner on different distributions over the training samples.

Fig. 2 is an overview of our approach. Given a training set S consisting of m training samples $\{(f_1, \mathbf{r}_1), \ldots, (f_m, \mathbf{r}_m)\}$, our approach learns a group of diverse SMT systems on different distributions $\{D_t(i) \mid t = 1, \ldots, T\}$ over S, where $D_t(i)$ denotes the *sample weight* on the i-th training sample. The key point here is that our approach maintains a set of sample weights $\{D_t(0), \ldots, D_t(m)\}$ and adjusts them after each weak system is learned with the base learning algorithm. Initially all the samples have equal weights. A succession of weak systems are then generated iteratively by: (1) training a new weak system $u(\boldsymbol{\lambda}_t^*)$ using D_t; (2) and generating a new distribution D_{t+1} using Bagging or Boosting-based weight updating methods. In this article these two procedures are called *training* and *re-weighting*, respectively. Finally, a strong system is built from $\{u(\boldsymbol{\lambda}_i^*)\}$ that are obtained in the previous steps.

Obviously, the approach presented in Fig. 2 follows the general framework of Bagging and Boosting [10,35,37,52,82]. However, both algorithms are originally designed for classification tasks that differs greatly from machine translation in natural language processing. When applying Bagging and Boosting to SMT system combination, we need to reconsider the following fundamental issues:

- *Member system generation: How should each weak system (or member system) be induced?* In traditional SMT paradigm, all training samples are equally weighted and the SMT training is not influenced by sample weights. Improved methods are required for generating diversified weak systems using different distributions over the training set.
- *Sample re-weighting: How should each distribution be chosen on each round?* In our approach, the sample distribution is required to be changed for generating different weak systems. Thus we need to investigate methods to adjust the weights of certain samples according to some criteria, such as criteria that concentrate more on the samples with relatively low translation quality.
- *Strong system generation: How should the weak systems (or member systems) be combined to form a strong system?* Differing from classification systems, SMT systems do something like ranking a list of translation candidates (or a translation lattice).[4] It is not so straightforward to combine the k-best outputs by reusing the simple voting strategy adopted in

[3] We will give a discussion on this issue in Section 6.4.

[4] In traditional classification tasks [7], the possible labels for a given input can be enumerated efficiently. Therefore, the combination of classification systems is very trivial: given an input sample, we score each label by individual systems, and simply select the highest-scoring one as the output by voting. However, in machine translation, it is intractable to generate all potential translations due to the extremely high complexity in decoding [54]. Instead, the system combination is performed on a limited sub-space of all translations (such as k-best translations and translation lattice) produced by member systems, which results in complex combination schemes.

classifier combination. So we also need to consider how to select or generate the final translation from the ensemble of k-best outputs produced by member systems.

To fit Bagging and Boosting into SMT system combination, all three of the above issues need to be addressed. In the rest of this section, we describe our solutions in detail.

3.2. Base learning algorithm

To optimize the feature weights, we choose Minimum Error Rate Training (MERT), an optimization algorithm introduced by Och [79], as the base learning algorithm in this work. The basic idea of MERT is to search for the optimal weights by minimizing a given error metric on the training set, or in other words, maximizing a given translation quality metric. Let $F = f_1 \dots f_m$ be m source sentences, $u(\boldsymbol{\lambda})$ be an SMT system, $E(u(\boldsymbol{\lambda})) = e_1^* \dots e_m^*$ be the translations produced by $u(\boldsymbol{\lambda})$, and $R = \boldsymbol{r}_1 \dots \boldsymbol{r}_m$ be the reference translations where $\boldsymbol{r}_i = \{r_{i1}, \dots, r_{iN}\}$. The objective of MERT can be defined as:

$$\lambda^* = \arg\min_{\lambda} \mathbf{Err}\big(E\big(u(\lambda)\big), R\big) \tag{3}$$

where **Err** is an *error rate* function. Generally, **Err** is defined with an automatic metric that measures the number of errors in $E(u(\boldsymbol{\lambda}))$ with respect to the reference translations R. Since any evaluation criterion can be used to define **Err**, MERT can seek a tighter connection between the feature weights and the translation quality. However, involving MT evaluation metrics generally results in an unsmoothed error surface, which makes the straightforward solution of Eq. (3) not trivial. To address this issue, Och [79] developed a grid-based line search algorithm to approximately solve Eq. (3) by performing a series of one-dimensional optimizations of the feature weight vector, even if **Err** is a discontinuous and non-differentiable function. While Och's method cannot guarantee to find the global optima, it has been recognized as a standard solution to learning feature weights for current SMT systems due to its simplicity and effectiveness.

Like most state-of-the-art SMT systems [15,57], in this work we choose BLEU [84] as the accuracy measure to define the error function used in MERT.[5] However, the method described above cannot work in our case because the learning procedure is not sensitive to the distribution $D_t(i)$. That is, we cannot use MERT to generate different SMT systems on different distributions $\{D_t(i)\}$ by optimizing standard BLEU scores directly. To handle this problem, we modify the original definition of BLEU so that the error is measured with respect to the distribution $D_t(i)$ and MERT can accept weighted training samples accordingly. In this article the modified version of BLEU is called *weighted BLEU* (WBLEU). It has the following form[6]:

$$\mathbf{WBLEU}\big(E\big(u(\lambda)\big), R\big) = \mathbf{BP}\big(E\big(u(\lambda)\big), R\big) \times \prod_{q=1}^{4} \mathbf{Precision}_q\big(E\big(u(\lambda)\big), R\big)^{1/4} \tag{4}$$

where

$$\mathbf{BP}\big(E\big(u(\lambda)\big), R\big) = \exp\left(1 - \max\left\{1, \frac{\sum_{i=1}^{m} D_t(i) \min_{1 \leqslant j \leqslant N}\{|g_1(r_{ij})|\}}{\sum_{i=1}^{m} D_t(i)|g_1(e_i^*)|}\right\}\right) \tag{5}$$

$$\mathbf{Precision}_q\big(E\big(u(\lambda)\big), R\big) = \frac{\sum_{i=1}^{m} D_t(i)|g_q(e_i^*) \cap (\bigcup_{j=1}^{N} g_q(r_{ij}))|}{\sum_{i=1}^{m} D_t(i)|g_q(e_i^*)|} \tag{6}$$

Here $g_n(s)$ is the multi-set[7] of all n-grams in a string s. In the above definition, $\mathbf{BP}(E(u(\boldsymbol{\lambda})), R)$ is the *brevity penalty*. $\sum_{i=1}^{m} D_t(i) \min_{1 \leqslant j \leqslant N}\{|g_1(r_{ij})|\}$ and $\sum_{i=1}^{m} D_t(i)|g_1(e_i^*)|$ are the effective reference length and the length of MT output over the weighted training samples, respectively. According to this formulation, the translation would be penalized when its length is shorter than the effective reference length.

The second factor on the right-hand side of Eq. (4) indicates the *n-gram precision*. Here $\sum_{i=1}^{m} D_t(i)|g_q(e_i^*)|$ is the number of n-grams in the MT output, and $\sum_{i=1}^{m} D_t(i)|g_q(e_i^*) \cap (\bigcup_{j=1}^{N} g_q(r_{ij}))|$ counts the clipping presents of n-grams in the reference translations. As all the n-grams in e_i^* and $\{r_{ij}\}$ are weighted by $D_t(i)$, the n-gram precision is affected more by the samples with large $D_t(i)$. In other words, if a training sample has a larger weight, the corresponding n-grams will have more impact on the overall score $\mathbf{WBLEU}(E, R)$ and the training sample will gain more importance in MERT.

Obviously, the original definition of BLEU is a special case of WBLEU when all the training samples are equally weighted. This also means that the training on the first round (i.e., $t = 1$) actually makes no difference from the traditional SMT training. So the first weak system $u(\boldsymbol{\lambda}_1^*)$ can be regarded as the *baseline* system generated by our approach.

---

[5] Although BLEU is used for our study, our approach is applicable to other popular metrics, such as NIST score [24] and mWER [78].

[6] Here we use the NIST-version BLEU where the *effective reference length* is the length of the shortest reference translation.

[7] We follow [16] to define the multi-sets: given a multi-set X, we use $\#_X(a)$ to denote the number of times a appears in X. Then:

$$|X| = \sum_a \#_X(a), \qquad \#_{X \cap Y}(a) = \min\big(\#_X(a), \#_Y(a)\big), \qquad \#_{X \cup Y}(a) = \max\big(\#_X(a), \#_Y(a)\big)$$

As the weighted BLEU is used to measure the translation accuracy on the training set, the error rate function is defined to be:

$$\mathbf{Err}\big(E\big(u(\lambda)\big), R\big) = 1 - \mathbf{WBLEU}\big(E\big(u(\lambda)\big), R\big) \tag{7}$$

3.3. Re-weighting

Once a weak system is learned, our approach updates the weights of samples so that it can focus more on a different distribution in the next round of system training. This procedure takes a weak system and a distribution of training set as input and generates a new distribution for training the following system. To adjust the distribution over training samples, we use two methods that are inspired by two types of algorithms: those that adjust the distribution of the training set without knowing the performance of previous weak system (as in Bagging), and those that adaptively adjust the distribution based on the performance of previous weak system (as in Boosting).

3.3.1. Bagging-based Re-weighting

Bagging, or *Bootstrap aggregating*, is one of the most effective and simplest methods in ensemble learning [10]. As a special case of model averaging, it promotes model variance by using different versions of a bootstrapped training set, i.e., sampling training set with replacement. Each version of the training set is then used to train a different model. Although Bagging is originally designed for classification and usually used to improve decision tree models [10,11,25], it has been successfully applied to many types of models of classification or regression, and has been recognized as one of the state-of-the-art ensemble learning algorithms. In this work we apply Bagging to sample re-weighting. The following pseudo-code describes the Bagging-based re-weighting algorithm for our approach.

Bagging-based Re-weighting Algorithm

```
Input: a training set S of m samples
Output: a set of sample weights {D_t(i) | i = 1, ..., m} over S
1  Function BaggingBasedReweighting (S)
2    for i = 1 to m do
3      D_t(i) = 0
4    m' = m × τ                          ◁ determine the rounds of sampling
5    for j = 1 to m' do                  ◁ sampling for m' times
6      i = DrawSample(1, m)              ◁ draw a sample with replacement
7      D_t(i) = D_t(i) + 1/m'            ◁ re-weighting
8    return {D_t(i)}
9  Function DrawSample(1, m)
10   return a random number between 1 and m
```

Given a training set S consisting of m samples, the algorithm produces a distribution over S by drawing m' samples uniformly from S. If a sample with index i is drawn from S, $D_t(i)$ will be updated by adding a number of $1/m'$. Finally the resulting sample weight has a form of $D_t(i) = x/m'$, where x is an integer ranging from 0 to m'. $x = 0$ means that the sample is not selected as the training sample and will not affect the training of following weak system. τ is a real-valued parameter that controls the number of active samples in the training set. In this work it chooses values in the range of $(0, 1]$.

Theoretically, Bagging is effective only if the base learning algorithm is *unstable* [11], that is, a small change in the training set can result in large changes in the learned weak system. The *instability* of learning algorithms has been studied in classification [11,12]. In this article, we discuss the issue in the context of SMT. Instead of presenting a theoretical analysis, we give an empirical study on this issue in the evaluation section (Section 4.7), followed by a further discussion in Section 6.3.

3.3.2. Boosting-based Re-weighting

Boosting is a method that is introduced by Schapire [91] for boosting the performance of weak learning algorithms. Like Bagging, Boosting adjusts the sample weights to perform weak system training on different distributions over training set. Beyond this, Boosting differs sequentially from Bagging, since it changes the sample weights based on the weak system generated previously. The basic idea of Boosting is straightforward: it places the most weight on the samples on which the preceding weak systems perform poorly, and thus forces the weak system learner to focus on the "harder" samples. Boosting has several advantages [92]: it is simple and easy to implement; it requires no prior knowledge about the base learning algorithm; and it has very nice theoretical properties, such as the good bounds on both training error and generalization error [29,72,73]. Due to these advantages, we choose Boosting as another re-weighting method in this work. The following pseudo-code shows the Boosting-based re-weighting algorithm for our approach.

Boosting-based Re-weighting Algorithm

Input: a training set S of m samples, current sample weights $\{D_t(i) \mid i = 1, \ldots, m\}$, the newly-generated weak system $u(\boldsymbol{\lambda}_t^*)$.
Output: a new set of sample weights $\{D_{t+1}(i) \mid i = 1, \ldots, m\}$ over S

1 **Function** BoostingBasedReweighting (S, $\{D_t(i)\}$, $u(\boldsymbol{\lambda}_t^*)$)
2 $\varepsilon = \text{GetErr}\ (u(\boldsymbol{\lambda}_t^*), S)$
3 **set**

$$\alpha = \frac{1}{2}\ln\left(\frac{1+\varepsilon}{\varepsilon}\right) \tag{8}$$

4 **for** $i = 1$ **to** m **do**
5 **update**

$$D_{t+1}(i) = \frac{D_t(i) \cdot \exp(\alpha \cdot loss(i))}{Z_t} \quad \triangleleft \text{ re-weighting} \tag{9}$$

where $loss(i)$ is the loss on the i-th sample, and Z_t is the normalization factor
6 **return** $\{D_{t+1}(i)\}$
7 **Function** GetErr ($u(\boldsymbol{\lambda}^*)$, S)
8 **return** error rate of $u(\boldsymbol{\lambda}^*)$ on S

This algorithm increases the weights of the samples that are relatively poorly translated by the current weak system so that the MERT-based learner can focus on hard samples in next round. A larger updated weight $D_{t+1}(i)$ indicates that the corresponding sample impacts more on the training of following weak system. Equation (9) shows the update rule with two parameters α and $loss(i)$ in it.

The main effect of α is to scale the weight updating (e.g., a larger α means a greater update). Once a weak system is received, the algorithm calculates the corresponding error rate ε (line 2) and chooses the parameter α as in Eq. (8). Intuitively, α can be regarded as a measure of the importance that the t-th weak system gains in Boosting. As is defined in Eq. (8), α gets larger as ε gets smaller, and always has a positive value.[8] In this way, a weak system with a smaller error rate would affect more on the weight updating.

$loss(i)$ is the loss on the i-th sample. For each i, let $\{e_{i1}, \ldots, e_{ik}\}$ be the k-best translation candidates produced by the current weak system. The loss function is defined to be:

$$loss(i) = \text{BLEU}(\hat{e}_i, \mathbf{r}_i) - \frac{1}{p}\sum_{j=1}^{p}\text{BLEU}(e_{ij}, \mathbf{r}_i) \tag{10}$$

where $\text{BLEU}(e, \boldsymbol{r})$ is a function that returns the smoothed sentence-level BLEU score [65] of the translation e with respect to the reference translations $\boldsymbol{r}$, and $\hat{e}_i$ is the oracle translation which is selected from $\{e_{i1}, \ldots, e_{ik}\}$ in terms of $\text{BLEU}(e_{ij}, \mathbf{r}_i)$. $\text{BLEU}(\hat{e}_i, \mathbf{r}_i)$ is the oracle BLEU score and indicates the upper-bound performance of the system on the i-th sample. $\sum_{j=1}^{p}\text{BLEU}(e_{ij}, \mathbf{r}_i)/p$ is the average BLEU score of the top-p translation candidates generated by the MT system. Here p is a parameter that controls how many translation candidates are involved in calculating the loss. Intuitively, Eq. (10) defines a "distance" of the MT output from the oracle translation. A large "distance" means a large cost that we guess the top-p translation candidates instead of the oracle translation. By integrating $loss(i)$ into Eq. (9), $loss(i)$'s value accounts for the magnitude of weight update, that is, $D_t(i)$ gets a large update when the current weak system produces poor translations for the i-th sample. Note that the definition of the loss function here is similar to that used in [17] where only the top-1 translation candidate (i.e., $p = 1$) is taken into account.

3.4. Strong system generation

In the last step of our approach, a strong translation system is built from the ensemble of weak systems. This procedure is exactly the same as traditional system combination, i.e., given a set of translation candidates generated by multiple systems, we select the "best" candidate as the final output or generate a new translation that are "better" than any of them. Therefore, all of current system combination methods are applicable to the strong system generation in our case. In this work we consider three types of methods for learning the final translation from the outputs of member systems:

- *Sentence-level combination*. Sentence-level combination (or *hypothesis selection*) is one of the simplest instances of system combination, and has been successfully employed in several MT systems [26,49]. Its idea is straightforward: given a pool of translation candidates, the final translation is generated by choosing the most promising candidate from the pool according to certain criteria. For example, we can select the candidate that agrees most with other candidates by using n-gram matching scores [49]. Also, Minimum Bayes-Risk (MBR) decoding [41] is applicable to our case for selecting the minimum risk translation of k-best list generated by a single translation engine [60] or multiple translation engines [42].

[8] Note that the definition of α here is different from that in the original AdaBoost algorithm [37,92] where α is a negative number when $\varepsilon > 0.5$.

- *Word-level/phrase-level combination*. Unlike sentence-level combination, word-level/phrase-level combination can generate new translations differing from any of the original translations. While many word-level/phrase-level combination methods exist, we choose confusion networks in this work due to their superiority in recent system combination tasks [6,76,89]. In system combination based on confusion networks, a primary translation is first chosen as a backbone (or skeleton) and aligned with other translation candidates. Then, a new search space is constructed from those backbone-aligned translation candidates. The final consensus translation is generated via a voting procedure of a feature-based model. In this work we take confusion network-based combination as a more sophisticated alternative to sentence-level combination.
- *Re-ranking*. Re-ranking is a very popular approach to improving the performance of single translation engines. Basically, re-ranking can be cast as a special instance of multi-pass decoding in SMT [56]. In the first pass, a set of translation candidates are generated using individual models and local features. Then, in the second pass, these translation candidates are ranked according to an additional model with the use of more global features. Though re-ranking is originally developed to re-rank the k-best list generated by a single MT system, it can work on selecting the best translation from multiple k-best lists in our case because all the member systems generated by Bagging/Boosting are based on the same model and feature set. Here we choose re-ranking as an additional method for final system generation. Note that re-ranking is of no use when multiple structurally different systems are provided, and thus cannot be employed to combine systems based on different base models.

4. Experiments

Our experiments are conducted on the NIST Chinese–English translation tasks[9] in three SMT systems.

4.1. Base systems

The first system is a *phrase-based* system with two reordering models, including the maximum entropy-based lexicalized reordering model proposed in [110] and the hierarchical phrase reordering model proposed in [40]. In addition to the reordering features, we use most standard features adopted in current state-of-the-art phrase-based systems [57,58], including bidirectional phrase translation probabilities, bidirectional lexical weights, an n-gram language model, target word penalty and phrase penalty. In our system all phrase pairs are limited to have source length of at most 3, and the reordering limit is set to 8 by default.[10]

The second system is an in-house reimplementation of the *Hiero* system which is based on the hierarchical phrase-based model proposed by Chiang [14]. The basic feature set used in our implementation is designed according to [15].

The third system is a *syntax-based* system based on the string-to-tree model [39,70]. In this system, both minimal GHKM and SPMT rules are extracted from the bilingual text, and larger rules are generated by composing two or three minimal GHKM and SPMT rules. The feature design is mainly based on a state-of-the-art MT system described in [70]. For example, we use the bidirectional lexical and phrase-based translation probabilities (4 features), a syntactic feature – root normalized conditional probability, an n-gram language model, target word penalty, rule penalty and three binary features – lexicalized rule; low-frequency rule; composed rule. For integrating n-gram language model into decoding efficiently, rules containing more than two variables are binarized using the synchronous binarization method [108,112].

All three of the above systems are built using the NiuTrans open-source SMT toolkit [107]. To achieve state-of-the-art performance, we further advance these systems in three ways: (1) we use an additional feature that counts the high-precision lexicon mappings involved in each translation derivation[11]; (2) we add a word deletion feature so that the systems can learn how often word deletion is performed; (3) and we use four specialized translation modules to translate date, time, name and byline respectively, and insert their translations into the SMT systems.

To obtain baseline performance for comparison, we use the system produced by our approach when the number of iterations (i.e., T) is set to 1. We train each system for 5 times using MERT with different initial values of feature weights to generate a group of *baseline candidates*, and then select the best-performing one from the group as the final baseline system (i.e., the starting point in Bagging and Boosting) for the following experiments.

4.2. System combination methods

As stated in Section 3.4, the strong system generation is doing something rather similar to system combination, and many methods are available for learning the final translations from the output ensemble of member systems. On the other hand, our approach is inspired by traditional system combination, and it is well worth a comparison of our Bagging/Boosting-based approaches and other counterparts in the general SMT combination framework. In our experiments we choose several system combination and re-ranking methods to build (1) baselines in our experiments (Sections 4.4, 4.10 and 4.12); and

[9] http://www.itl.nist.gov/iad/mig//tests/mt/.

[10] Our in-house experimental results show that this system performs slightly better than Moses [57] on Chinese–English translation tasks.

[11] In this work the high-precision lexicon mapping is identified by using the Chinese–English Translation Lexicon Version 3.0 (LDC catalog number: LDC2002L27).

(2) experiment with various alternatives of strong system generation (Section 4.11). In the following, we present more details about the system combination and re-ranking methods used in our experiments.

- *Default method*. A sentence-level combination method is chosen as the default method of strong system generation in our proposed framework. Let H_{all} be the union set of the k-best translations of member systems. This method selects the final translation based on the following scoring function

$$e^* = \arg\max_{e \in H_{all}} \sum_{t=1}^{T} \beta_t \cdot \phi_t(e) + \psi(e, H_{all}) \tag{11}$$

  where $\phi_t(e)$ is the log-scaled model score of e in the t-th member system, and β_t is the corresponding feature weight which implies the importance of the t-th member system. $\psi(e, H_{all})$ is a consensus-based scoring function which has been successfully adopted in SMT system combination [26,49,63]. The computation of $\psi(e, H_{all})$ is based on a linear combination of a set of n-gram consensuses-based features, namely *n-gram agreement and disagreement features*. In this work the n-gram agreement and disagreement features are exactly the same as those used in [26], and range from unigram to 4-gram. Since all the features in the above model are combined linearly, we use MERT to optimize the associated feature weights on the same data set used for learning member systems.[12] The major advantage of sentence-level combination is its simplicity and effectiveness. For example, previous work has demonstrated that, though very simple, it can achieve comparable performance with more sophisticated methods on several tasks [26]. Hence we choose it as the default solution to the final translation generation of our approach in the following experiments.
- *Minimum Bayes-Risk* (*MBR*) *decoding*. MBR is another popular technique that is useful in selecting the most promising translation from an ensemble of translation candidates. The MBR approach is originally developed for automatic speech recognition [41] and has recently been applied to SMT [60,111]. The basic idea of MBR is to seek the best translation with the least expected loss under a probabilistic model [9]. It provides a very simple way to consider various error metrics in selecting the best translation from k-best outputs. For example, we can incorporate the error criterion (e.g., BLEU) into the loss function and make use of n-gram agreement between different candidates during decoding. In our experimental comparison, we follow the system presented in [60] to build our MBR baseline. We use the BLEU-inspired loss proposed in [103] to implement the loss function. We choose this loss function because it has been successfully applied to several state-of-the-art SMT systems [61,103].
- *System combination using confusion network and indirect HMM* (*IHMM*) *alignment*. The third combination method is based on confusion networks. It builds confusion networks using the indirect HMM alignment model presented in [46]. Unlike traditional HMMs that are trained via Maximum Likelihood Estimation (MLE), IHMMs learn their parameters indirectly from a variety of sources, such as word surface/semantic similarity and distance-based distortion penalty. By using a similarity model for synonym matching and a distortion model for word ordering, this approach shows state-of-the-art performance on the NIST Chinese–English translation tasks, even significantly outperforms its counterpart that uses the Translation Error Rate (TER)-based alignment [46]. In our experiment, we adopt the same setting as that used in [46]. The skeleton is selected using MBR, where TER [102] is used as the loss function. All hypotheses are assigned a uniform posteriori probability. For confusion network decoding, word posteriors, a 5-gram language model score and target word penalty are used as features and combined in a log-linear fashion.
- *System combination using confusion network and METEOR alignment*. We also consider another combination method that builds confusion networks using the METEOR alignment. This method has been implemented in an open-source toolkit [47] which is one of the most successful combination systems in WMT shared tasks.[13] Instead of aligning to a single skeleton, it treats single best outputs from each system symmetrically. The METEOR aligner [4] is used to align 1-best translations from each individual system. In our experiment, we run Heafield and Lavie's system in its default setting.[14]
- *Boosted re-ranking*. As discussed in Section 3.4, re-ranking is also a good choice for generating the final translation from the k-best outputs of member systems. One of the related approaches we consider here is the boosted re-ranking approach proposed by Duh and Kirchhoff [30]. They use Boosting to improve their k-best re-ranking result without any additional features. Unlike our approach, they do not generate diverse translation candidates using Bagging or Boosting, but directly learn an ensemble of re-rankers from a (fixed) k-best output of a single SMT system. As their

[12] Note that the total number of features used in this model may be relatively large for MERT. For example, when 30 member systems are provided, there are 38 features in total (30 members systems plus 8 n-gram agreement and disagreement features). However, it is well known that MERT does not work well if a large number of features are involved in optimization [17]. To address this issue, we modify the original MERT program with some tricks. The basic idea is that we divide the features into a few groups of features, and optimize each feature group at a time rather than optimizing those features all together. For example, in our implementation, we first optimize the n-gram agreement and disagreement features. Then, we divide the remaining features into three groups (with ten features in each group), and call MERT to optimize each group of features respectively. Once each feature group is optimized, we repeat this process for a number of rounds until the change of feature weights is below a pre-defined threshold. In this way, we keep the number of features in each MERT run at a reasonable level. Though it is more time-consuming, this method works well in our sentence-level combination system and obtains a very stable convergence.

[13] http://www.statmt.org/wmt10/.

[14] http://kheafield.com/code/memt/.

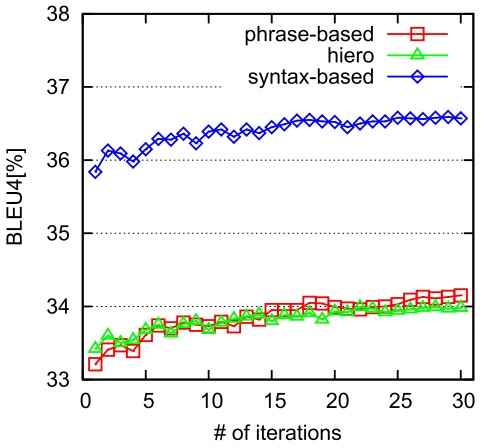

Fig. 3. BLEU score of Bagging on the development set.

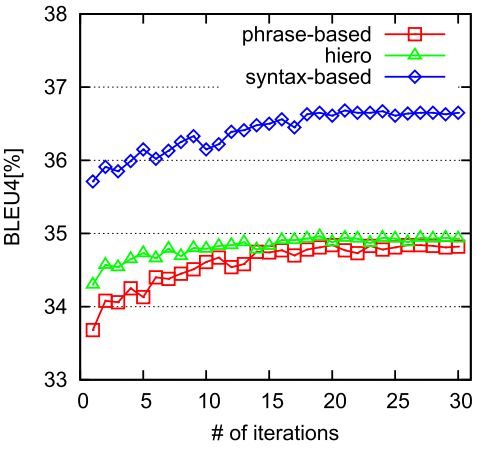

Fig. 4. BLEU score of Bagging on the test set of MT04.

approach requires k-best candidates generated by only one SMT system, its training and inference are very fast. So, in our experiment, we also implement Duh and Kirchhoff's approach for comparison.

- *Discriminative re-ranking*. Discriminative re-ranking is another straightforward solution to re-ranking k-best translations, and has been widely adopted in MT. In the general framework of discriminative re-ranking, we first choose a number of features to indicate whether a candidate should be ranked better than others. Then we employ a discriminative model to make use of these features and generate the final ranking result. In our implementation, we follow the approach presented in [97]. In addition to the features used in our baseline systems, we use three additional features for re-ranking, including IBM model 1, POS-based language models (bi-gram–5-gram), and matched parentheses/quotation marks. These features have been proved to be very effective in MT hypotheses re-ranking [80].

4.3. Experimental setup

Our bilingual data consists of 138K sentence pairs (3.2M Chinese words and 4.1M English words) extracted from the FBIS data set.[15] GIZA++[16] is employed to perform the bi-directional word alignment between the source and target sentences, and the final word alignment is generated using the grow-final-diag method. All the word-aligned sentence pairs are then used to extract phrase tables, synchronous grammars, and reordering models for the SMT systems. A 5-gram language model is built on the Xinhua portion of the English Gigaword corpus[17] in addition to the target-side of the bilingual data. Berkeley Parser[18] is used to generate the English parse trees for rule extraction of the syntax-based system. The data set used for weight training in our approach comes from the NIST MT03 evaluation set (919 sentences). To speed up MERT, all the sentences with more than 20 Chinese words are removed. The test sets are the NIST evaluation sets of MT04 (1788 sentences), MT05 (1082 sentences) and MT06 (1664 sentences). The translation quality is evaluated in terms of case-insensitive NIST version BLEU metric. Statistical significance test is conducted using the bootstrap re-sampling method proposed in [55].

For all three base systems described above, we use a CKY-style decoder with beam search and cube pruning [51] to decode new Chinese sentences. By default, both the beam size (or beam width) and the size of k-best list are set to 20.[19] For our Bagging/Boosting-based approach, the maximum number of iterations is set to 30, and p (Eq. (10)) is set to 5.

4.4. Evaluation of translations

First we investigate the effectiveness of our approach on improving single translation engines. Figs. 3–6 show the BLEU curves of the Bagging-based approach on the development and test sets for the three SMT systems, where the X-axis is the iteration number and the Y-axis is the BLEU score of the final system generated by our approach. The points at iteration 1 stand for the performance of the baseline systems. We see, first of all, that all three types of SMT systems are improved on the development set as well as on the test sets. For all these systems, over 0.5 BLEU improvements are achieved after 8 iterations, and the BLEU scores tend to converge to stable values after 20 iterations. This trend also holds in Figs. 7–10 where

[15] LDC catalog number: LDC2003E14.
[16] http://code.google.com/p/giza-pp.
[17] LDC catalog number: LDC2003T05.
[18] http://code.google.com/p/berkeleyparser/.
[19] In our implementation, beam size = 20 means that only the top-20 best hypotheses according to model score are kept and the rest are discarded on each search stage.

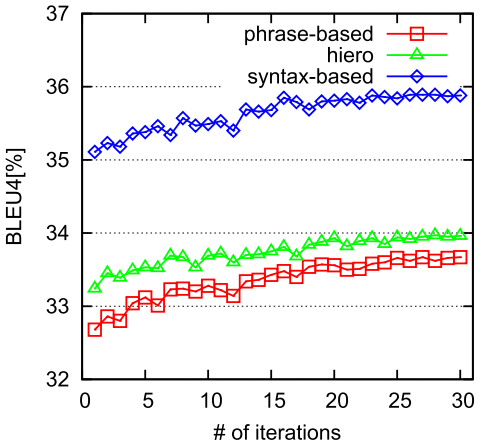

Fig. 5. BLEU score of Bagging on the test set of MT05.

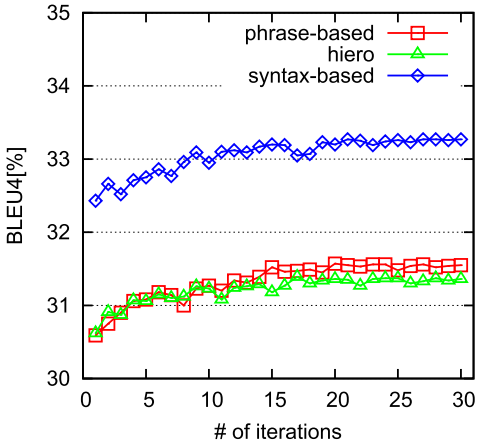

Fig. 6. BLEU score of Bagging on the test set of MT06.

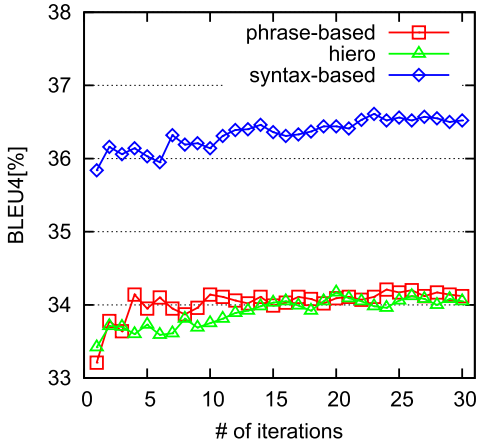

Fig. 7. BLEU score of Boosting on the development set.

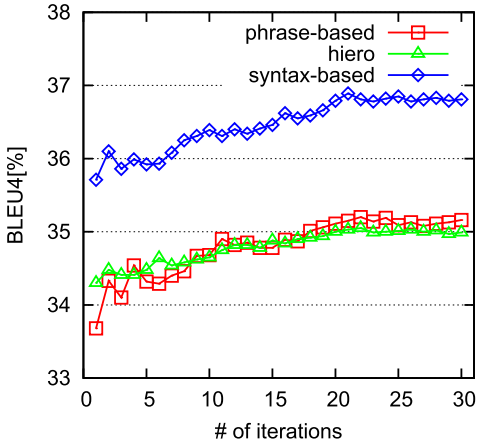

Fig. 8. BLEU score of Boosting on the test set of MT04.

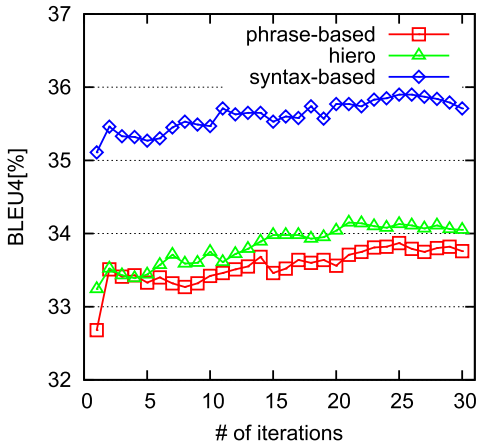

Fig. 9. BLEU score of Boosting on the test set of MT05.

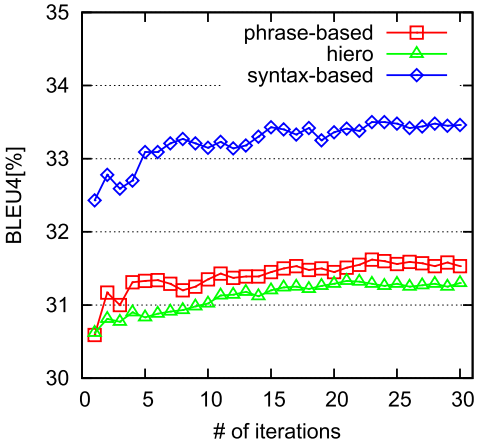

Fig. 10. BLEU score of Boosting on the test set of MT06.

the BLEU curves of the Boosting-based approach are plotted. More interestingly, it is observed that our approach seems to be more helpful to the phrase-based system than to the Hiero system and the syntax-based system. For the phrase-based system, it yields an improvement of over 0.6 BLEU points just after three iterations on all the data sets.

Tables 1–3 show more detailed evaluation results for our approach, where the BLEU scores at iteration 2, 5, 10, 15, 20 and 30 are reported. For comparison, we also report the results of various baselines:

Table 1
Summary of the evaluation results (BLEU4[%]) for the phrase-based system. + or ++ = significantly better than the first five baselines or all eight of the baselines (significance level is 0.05).

	BLEU4[%] (the phrase-based system)							
	MT03 (Dev)		MT04 (Test)		MT05 (Test)		MT06 (Test)	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
Baseline	33.21		33.68		32.68		30.59	
Baseline + *Fulldev*	33.02		33.89		32.83		31.00	
Baseline + 600*best*	33.32		33.93		32.84		30.76	
Baseline + *Batch*	33.54		33.88		32.77		30.92	
Baseline + *Timefirst*	33.37		33.70		32.90		30.68	
MBR	33.39		33.88		32.86		30.74	
Boosted Re-ranking	33.21		33.66		32.87		30.89	
Discriminative Re-ranking	33.13		33.76		32.91		30.58	
Our approach								
Iteration 2	33.41	33.78	34.08	34.33+	32.86	33.51++	30.75	31.17
Iteration 5	33.61	33.95	34.13	34.32+	33.12	33.33+	31.08	31.33
Iteration 10	33.73	34.14++	34.61++	34.68++	33.28	33.42+	31.27	31.35
Iteration 15	33.95	33.99+	34.74++	34.78++	33.43+	33.46++	31.52++	31.45++
Iteration 20	33.99++	34.09++	34.84++	35.11++	33.56++	33.56++	31.57++	31.45++
Iteration 30	34.11++	34.12++	34.82++	35.16++	33.67++	33.76++	31.55++	31.59++

Table 2
Summary of the evaluation results (BLEU4[%]) for the Hiero system. + or ++ = significantly better than the first five baselines or all eight of the baselines (significance level is 0.05).

	BLEU4[%] (the Hiero system)							
	MT03 (Dev)		MT04 (Test)		MT05 (Test)		MT06 (Test)	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
Baseline	33.42		34.30		33.24		30.62	
Baseline + *Fulldev*	33.35		34.43		33.45		30.99	
Baseline + 600*best*	33.48		34.46		33.39		30.75	
Baseline + *Batch*	33.46		34.51		33.34		30.90	
Baseline + *Timefirst*	33.50		34.30		33.30		30.77	
MBR	33.54		34.55		33.42		30.73	
Boosted Re-ranking	33.30		34.48		33.20		30.85	
Discriminative Re-ranking	33.33		34.19		33.37		30.90	
Our approach								
Iteration 2	33.60	33.71	34.57	34.48	33.45	33.52	30.91	30.81
Iteration 5	33.68	33.73	34.73	34.48	33.53	33.44	31.06	30.83
Iteration 10	33.68	33.75	34.79	34.65	33.69	33.75++	31.23	31.02
Iteration 15	33.80	34.03+	34.83	34.88	33.75	33.98++	31.18	31.20
Iteration 20	33.93+	34.17+	34.88	35.00+	33.93++	34.04++	31.36++	31.29++
Iteration 30	33.98+	34.05+	34.93+	34.99+	33.96++	34.05++	31.36++	31.30++

- ***Baseline***: the naive baseline which is generated in the first iteration of our approach.
- ***Baseline* + *Fulldev***: the baseline system whose feature weights are optimized on the full set of development data, instead of the sentences of constrained length ($\leqslant$20 words).
- ***Baseline* + 600*best***: the baseline system with the k-best list size of 600 which equals to the maximum number of translation candidates accessed in our approach (20-best outputs of 30 member systems).
- ***Baseline* + *Batch***: we run MERT for as many times as there are iterations of Bagging/Boosting (30 times in this experiment). We then treat each MERT run as a member system, and apply the proposed approach to those MERT runs. This method results in a baseline of running MERT in batch mode.
- ***Baseline* + *Timefirst***: the baseline system which runs in the same amount of time as the final systems generated by Bagging/Boosting.[20]
- ***MBR***: the MBR system which runs on the k-best list of 600 translation candidates.[21]
- ***Boosted Re-ranking***: the re-ranking system based on the approach presented in [30]. The iteration number of boosted re-ranking is set to 30, which is the same as that used in our approach.

[20] To do this, we enlarge the beam width until the execution time (or decoding time) is equal to or more than that of the system generated using Bagging/Boosting. This method results in a beam width of 450 for all three of the SMT engines in our experiments.

[21] Note that the k-best MBR method is used in our experiment, while recent studies have reported that the lattice-based method [61,103] is relatively more powerful in the MBR paradigm. Here we choose k-best MBR because all the systems in this set of experiments generate the final translation from the k-best list of translation candidates, and it is fair to compare these approaches in the same setting of k-best output of MT system. The issue of lattice-based MBR will be discussed in Section 7.

Table 3
Summary of the evaluation results (BLEU4[%]) for the syntax-based system. + or ++ = significantly better than the first five baselines or all eight of the baselines (significance level is 0.05).

	BLEU4[%] (the syntax-based system)							
	MT03 (Dev)		MT04 (Test)		MT05 (Test)		MT06 (Test)	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
Baseline	35.84		35.71		35.11		32.43	
Baseline + *Fulldev*	35.55		35.90		35.30		32.85	
Baseline + 600*best*	35.95		35.88		35.23		32.58	
Baseline + *Batch*	36.00		36.01		35.23		32.81	
Baseline + *Timefirst*	35.89		35.82		35.13		32.76	
MBR	35.99		35.91		35.35		32.53	
Boosted Re-ranking	35.80		35.71		35.28		32.49	
Discriminative Re-ranking	35.80		35.45		35.37		32.44	
Our approach								
Iteration 2	36.13	36.16	35.91	36.10	35.23	35.46	32.66	32.78
Iteration 5	36.15	36.03	36.15	35.92	35.38	35.27	32.75	33.09
Iteration 10	36.39	36.14	36.15	36.39	35.49	35.47	32.95	33.15
Iteration 15	36.45+	36.36	36.50++	36.46+	35.68	35.53	33.20++	33.43++
Iteration 20	36.52++	36.44+	36.61++	36.79++	35.81+	35.77+	33.20++	33.36+
Iteration 30	36.57++	36.52++	36.65++	36.81++	35.88++	35.71++	33.27++	33.46++

- ***Discriminative Re-ranking***: the discriminative re-ranking system that runs on the same *k*-best list used in *MBR* and *Baseline* + 600*best*.

As can be seen from Tables 1–3, our approach outperforms all eight of the baselines. In particular, we have the following observations:

(1) BLEU improvement over *Baseline*
We see that both the Bagging-based and Boosting-based approaches achieve statistical significant BLEU improvements after 15 iterations, and generally achieves the highest BLEU scores after 20 iterations. For the phrase-based system, over 0.7 BLEU improvements are obtained after 10 iterations. The largest BLEU improvement of the phrase-based system is over 1 BLEU point. These results show again that our method is relatively more effective for the phrase-based system than for the other two systems, and thus confirms the observation we got in Figs. 3–10.

(2) Bagging vs. Boosting
Also, Tables 1–3 make a comparison of the Bagging-based and Boosting-based approaches. We see that the Boosting-based approach outperforms the Bagging-based approach in most cases. This result indicates that the member system learner can benefit from knowing the performance of the previous member system. However, the performance difference between the two approaches is small (less than 0.2 BLEU points). This suggests an interesting fact that Bagging, though very simple, is still effective in improving single translation engines.

(3) Bagging/Boosting vs. *Baseline* + *Fulldev*
In a sense, *Baseline* + *Fulldev* can be regarded as another reasonable baseline since the setting of our approach (Bagging/Booting) should not affect MERT for the baseline systems which run fine with sentences of unconstrained length. In our experiment, we observe that scaling the development data to the full set of MT03 provides a more stable convergence for MERT. This benefits the baseline systems, especially on the test sets. For example, for all three of the SMT systems, *Baseline* + *Fulldev* obtains about +0.4 BLEU improvements over its counterpart *Baseline* on the test sets of MT06, and offers a stronger baseline performance for our comparison. Compared to *Baseline* + *Fulldev*, our approach still shows significant improvements on most of the data sets.

(4) Bagging/Boosting vs. *Baseline* + 600*best*
The result of *Baseline* + 600*best* shows that the access to larger *k*-best lists is useful in improving the baseline systems. However, the improvement achieved by *Baseline* + 600*best* is modest compared to that achieved by *Boosting-30Iterations*. This supports the fact that the SMT systems can benefit more from the diversified outputs of member systems than from larger *k*-best lists produced by a single system.

(5) Bagging/Boosting vs. *Baseline* + *Batch* and *Baseline* + *Timefirst*
Baseline + *Batch* and *Baseline* + *Timefirst* can be viewed as two enhanced baselines which make comparable efforts on the search problem as Bagging/Boosting. Both of them show improvements over *Baseline* on all the evaluation sets due to the fewer search errors they make during decoding. However, Bagging and Boosting still significantly outperforms *Baseline* + *Batch* and *Baseline* + *Timefirst*, which implies a stronger ability in reducing search errors. An interesting finding here is that the translation quality can be improved by combining different MERT runs on the same development set. This observation reflects some sort of training instability in SMT, and confirms the result reported in recent studies on MERT run selection and hypothesis testing [18].

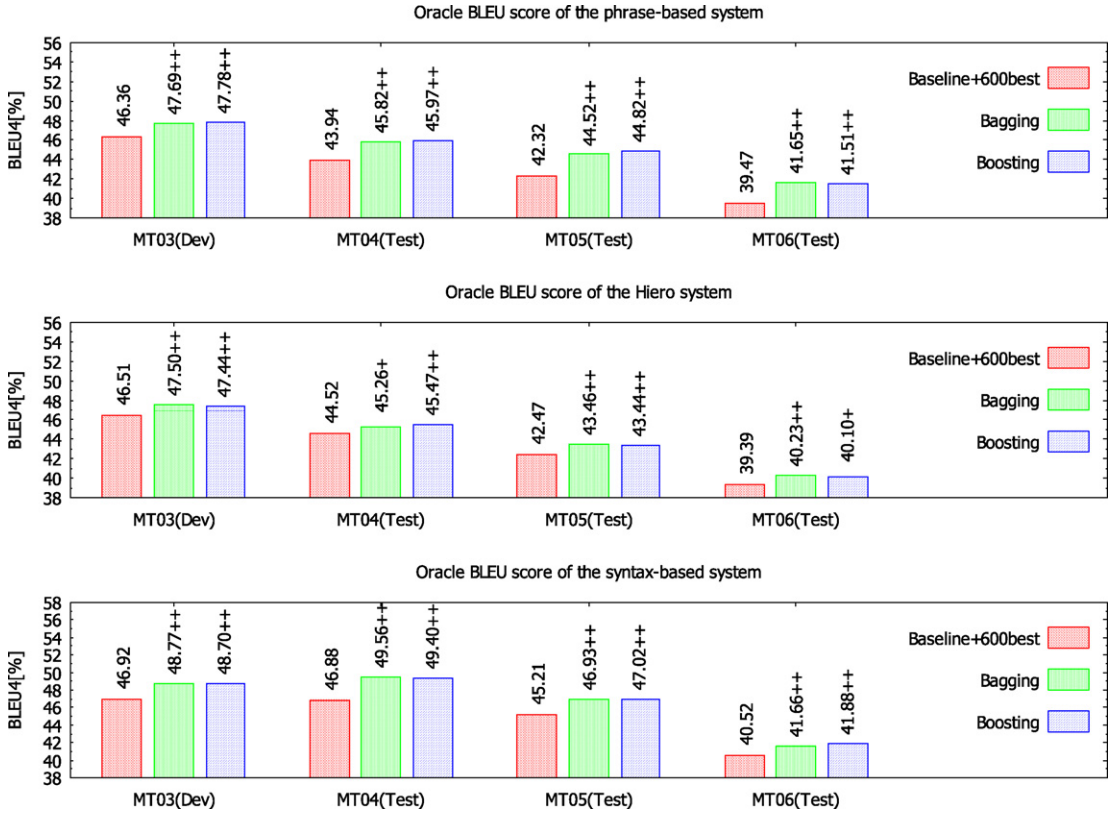

Fig. 11. Oracle BLEU scores of the phrase-based system, the Hiero system and the syntax-based system. + or ++ = significantly better than *baseline* + 600*best* (significance level is 0.05 or 0.01).

(6) Bagging/Boosting vs. MBR and re-ranking
The MBR and re-ranking methods are also reasonable baselines for our comparison because they are well known for their ability in improving single translation systems. As shown in the tables, though they outperform other baseline systems in some cases, the improvements are much less than those of the Bagging-based and Boosting-based approaches. These results support our argument that introducing diversified translations into candidate set is relatively more useful in improving single translation engines. Moreover, the comparison between *Boosted Re-ranking* and our approach draws an interesting conclusion that applying Boosting to re-ranking might be too late to access good-quality translations, and thus limits the capability of Boosting in MT. But this is essentially due to the shortcoming of re-ranking and would not be a problem of Boosting. Another interesting observation is that our default method of strong system generation and the MBR method seem to be more effective in improving our SMT systems, in comparison with re-ranking approaches. We attribute it to the use of consensus-based features which have been successfully used in system combination, but are generally absent in traditional re-ranking systems.

4.5. Evaluation of oracle translations

In the second set of experiments, we evaluate the oracle performance on the k-best lists of various systems, with a primary goal of studying the impact of our approach on upper-bound performance.

We choose *Baseline* + 600*best* as the baseline for comparison since it explores the equal number of translation candidates as that of our approach. Fig. 11 shows the result, where *Bagging* and *Boosting* stand for the ensemble of translation candidates generated by the Bagging-based approach and Boosting-based approach, respectively. As expected, the oracle performance of *Bagging* and *Boosting* is significantly higher than that of *Baseline* + 600*best*. This result indicates that our approach provides "better" translation candidates for final translation selection than enlarging the size of k-best list naively. It also gives us a rational explanation for the significant improvements achieved by our approaches, as shown in Section 4.4. In addition, it is observed that the oracle performance of *Bagging* and *Boosting* is very close to each other. This observation agrees with the previous result in Tables 1–3 that there is no significant difference in BLEU scores of the two approaches.

4.6. Diversity among member systems

To examine the effect of our approach on generating diversified member systems, we also study the change of diversities among the outputs of member systems. To measure diversity, we choose the TER metric proposed in [102]. A higher TER

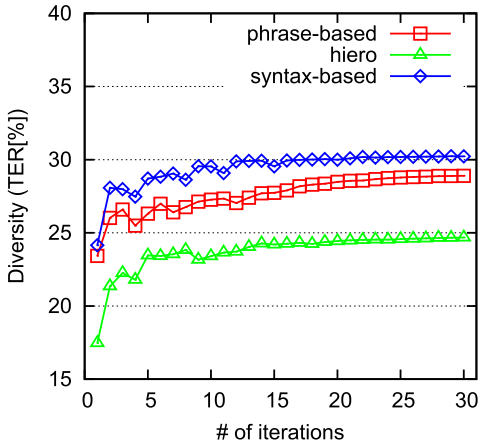

Fig. 12. Diversity (TER score) among member systems (Bagging).

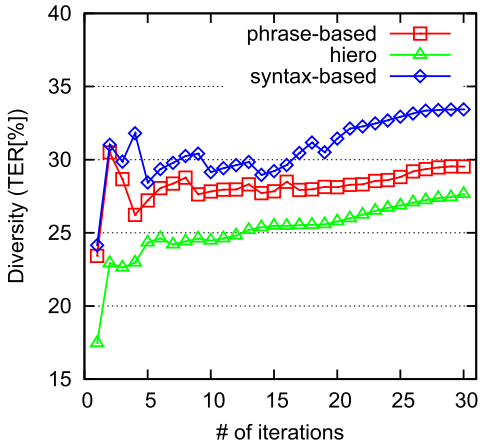

Fig. 13. Diversity (TER score) among member systems (Boosting).

score means that more edit operations are required if we transform one translation output into another, and thus reflects a larger diversity between the two outputs. Given a group of member systems, the TER score of this group is calculated by averaging the TER scores between the outputs of each pair of member systems in the group.

Figs. 12 and 13 show the curves of diversity on the development set, where the X-axis is the iteration number, and the Y-axis is the diversity. The points at iteration 1 stand for the diversities of baseline systems. In this work, the baseline's diversity is the TER score of the group of baseline candidates that are generated in advance (Section 4.1).

We see that in most cases the diversities increase as more iterations are performed, though they decrease a little at a few points. This result indicates that our approach is very useful in generating diversified member systems. Moreover, the diversities of baseline systems (iteration 1) are much lower than those of the systems generated in iterations 2–30. Together with the results shown in Figs. 3–10, it confirms the conclusion that the diversified translation outputs can lead to BLEU improvements over the baseline systems. In addition, it is observed that the Boosting-based approach leads to a larger volatile of diversity than that of the Bagging-based approach, especially in the first few iterations. The reason for this phenomenon might be that, compared to Bagging, Boosting generates relatively more diversified member systems due to its larger changes to the distribution of training set.

Also as shown in Figs. 12 and 13, the diversity of the Hiero system is much lower than those of the phrase-based system and the syntax-based system. This interesting observation indicates that, for the Hiero system, our approach generates fewer new translations than the other two types of systems. The relative lack of diversity in the Hiero system might be due to the *spurious ambiguity* in Hiero derivations which generally results in very few different translations in MT outputs [15].

Although TER score provides some information about the differences between member systems, it does not answer how many *new* translations are introduced by each individual member system. Here new translations refer to the translations that are not produced by previous member systems. More precisely, for the i-th member system, new translations are those that are never seen in the outputs of previous $i-1$ member systems. If more new translations are introduced, the system can select the final translation from a larger pool of diverse candidates. Thus the number of new translations generated by a member system can be regarded as an indicator of its ability of introducing "diversity" into system combination.

Figs. 14 and 15 show the percentage of new translations in the k-best outputs of each member system. As seen from the curves, all the members contribute to introducing new translations, though the numbers keep reducing. This is true even when the iteration number is over 20, which indicates that the final system can make benefits from the continuous "new candidates" provided by the newly-generated member systems.

4.7. Individual contribution of member systems

In addition to the evaluation of the final systems generated by our approach, we also evaluate the performance of each individual member system to study the behavior of member system generation.

Figs. 16 and 17 show the BLEU scores on the development set for each member system, where the baseline systems are those generated in the first iteration (i.e., member system 1). First, we see that all the member systems (2–30) are worse than the baseline systems. It agrees with the design of our approach that member systems (2–30) are trained on different distributions of the training set, while baseline system is trained on the original version of the training set. Furthermore, we observe that over 70% of the member systems are significantly different from the baseline system in BLEU scores. This result indicates that adjusting the distribution over training set can lead to significantly different SMT systems, and thus reflects an *instability* of the base learning algorithm. Also, it supports the motivation of this work that a strong system can be generated by combining multiple weak systems, even these systems are worse than the baseline.

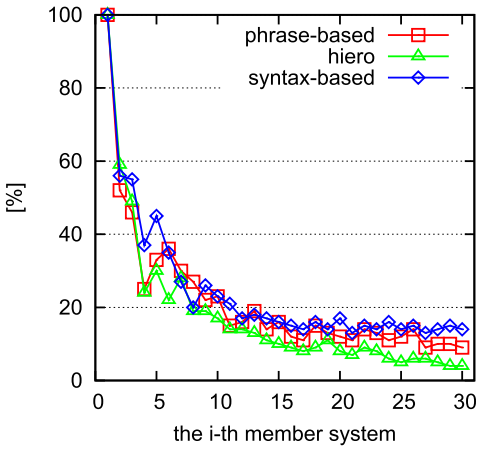

Fig. 14. New translations in the outputs of each individual member system (Bagging).

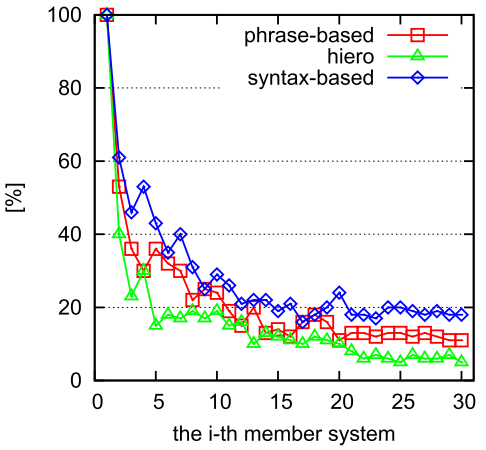

Fig. 15. New translations in the outputs of each individual member system (Boosting).

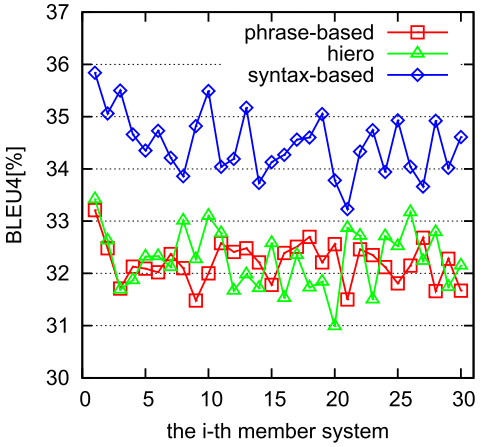

Fig. 16. BLEU score of each individual member system (Bagging).

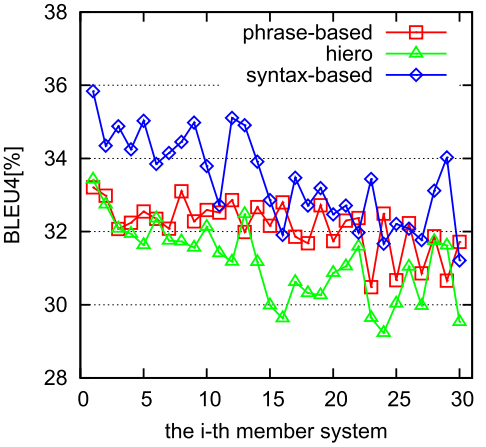

Fig. 17. BLEU score of each individual member system (Boosting).

Table 4
Summary of the evaluation results (BLEU4[%]) of member systems. *Average*, *Max* and *Min* stand for the average, maximum and minimum BLEU scores of the member systems 2–30. − or −− = significantly worse than baseline (significance level is 0.05 or 0.01).[a]

	BLEU4[%]					
	Phrase-based		Hiero		Syntax-based	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
Baseline (Member system 1)	33.21		33.42		35.84	
Member systems 2–30 (Average)	32.17−−	32.12−−	32.24−−	31.04−−	34.44−−	33.35−−
Member systems 2–30 (Max)	32.70	33.10	33.17	32.36−	35.50	35.11
Member systems 2–30 (Min)	31.48−−	30.48−−	30.99−−	29.23−−	33.23−−	31.21−−

[a] In the case of *Average*, the significance test is conducted on the member system whose BLEU score is closest to the average score.

Table 4 gives more statistics about the performance of member systems. As shown in the table, the average performance is 1-BLEU point lower than that of the baseline. Moreover, we observe that the BLEU variance of the Boosting-based approach is larger than that of the Bagging-based approach. A possible explanation for this phenomenon is that the Boosting-based approach results in relatively more changes in the distribution over training set, and consequently results in a larger variance among member systems.

We also investigate the question of how often the translations of each member system are selected as the final translations. Figs. 18 and 19 show the percentage of the final translations that come from each member system.[22] Basically, for

[22] Note that the sum of these numbers is not equal to 1 since a translation can be shared by more than one member systems.

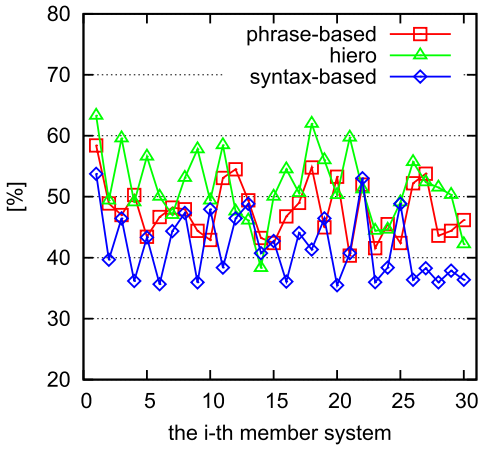

Fig. 18. Distribution of final translations over member systems (Bagging).

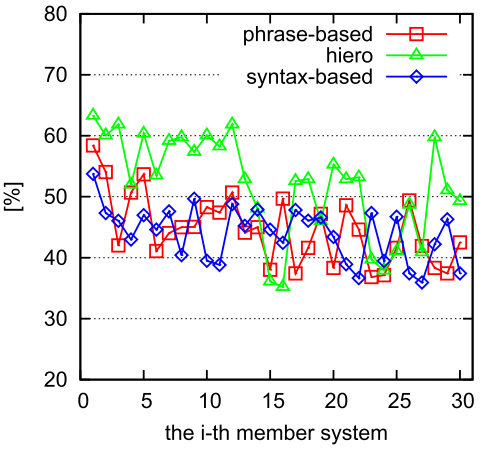

Fig. 19. Distribution of final translations over member systems (Boosting).

Table 5
Percentage of the translations that are generated by the final strong system and also covered by other systems. *Average*, *Max* and *Min* stand for the average, maximum and minimum number (in percentage) of final translations that can be found in the outputs of the member systems 2–30.

	% of final translations					
	Phrase-based		Hiero		Syntax-based	
	Bagging	Boosting	Bagging	Boosting	Bagging	Boosting
Baseline (Member system 1)	58.41%	59.45%	63.30%	64.80%	53.75%	54.79%
Member systems 2–30 (Average)	47.34%	44.19%	51.27%	51.63%	41.35%	43.62%
Member systems 2–30 (Max)	54.79%	54.04%	61.98%	61.85%	53.03%	49.66%
Member systems 2–30 (Min)	40.33%	36.83%	38.32%	35.20%	35.47%	35.93%

each member system, at least 30 percent of the translations are selected by the final system. From the detailed statistics in Table 5, we see that less than 60%, 65% and 55% of the final translations are covered by *Baseline* for the phrase-based system, the Hiero system and the syntax-based system, respectively. It means that our approach introduces over 40%, 35% and 45% of the *new* translations that are never seen in the three types of the base SMT systems.

4.8. Impact of k-best list size

In this work the generation of final translation relies on the k-best outputs of member systems. It is therefore worth investigating the properties of different sized k-best lists in the context of our approach. Figs. 20–22 show the BLEU scores of the Bagging-based approach at each individual setting of k-best output.[23] We see from the curves that all three types of the SMT systems benefit from larger k-best lists whose size is larger than 10. The major BLEU improvement appears when the k-best list size is set to around 20, while the improvement tends to converge when k is larger than 20. In the case of Boosting (Figs. 23–25), we observe a similar result – the "20-best" case outperforms the "5-best" and "10-best" counterparts; but a too large k-best list does not give further improvements. One possible reason for this lies in that low-rank translations introduce more noise into the pool of translation candidates for the final translation generation [44,49]. Especially, in our default method of strong system generation, the k-best list is used as an unweighted list of translation hypotheses (e.g., in Eq. (10) we uniformly average together all the candidates in the list), which somehow enlarges the influence of the noise. These results deliver a message that it might not be a good choice to advance our approach by enlarging k-best lists naively. In this work, setting the size of k-best list around 20 is enough to achieve satisfactory BLEU improvements.

4.9. Sensitivity analysis

We also study the behavior of our approach under different parameter settings. First, we adjust parameter τ to see the impact of the number of samples drawn from the original training set for the Bagging-based approach. Figs. 26–28 show the BLEU curves over iterations while τ is set to different values. We observe that the BLEU scores do not have significant changes as the value of τ changes. Even τ is set to 0.1 (i.e., selecting 10% of the samples in the original training set), our approach still achieves comparable performance with that in the setting of $\tau = 1$. This interesting result indicates that the

[23] To present the result in a more concise and clear way, we only plot the BLEU score for the odd number of iterations in the figures in Sections 4.8 and 4.9.

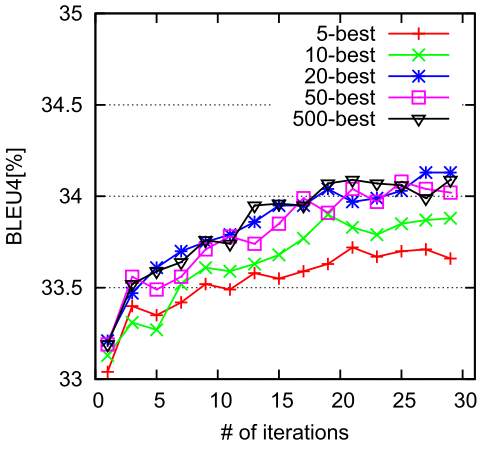

Fig. 20. BLEU curves with different sizes of k-best list (Bagging and phrase-based).

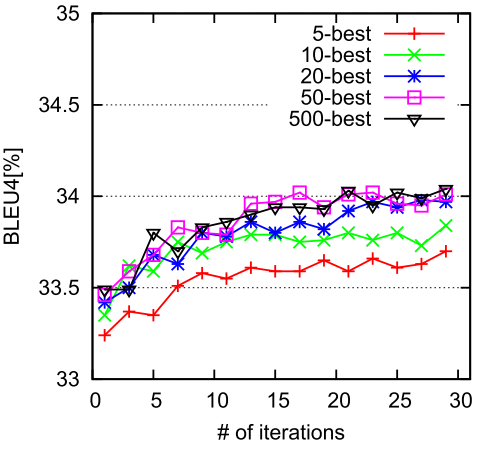

Fig. 21. BLEU curves with different sizes of k-best list (Bagging and Hiero).

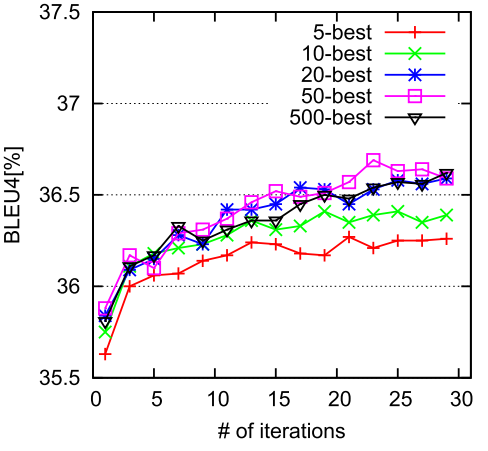

Fig. 22. BLEU curves with different sizes of k-best list (Bagging and syntax-based).

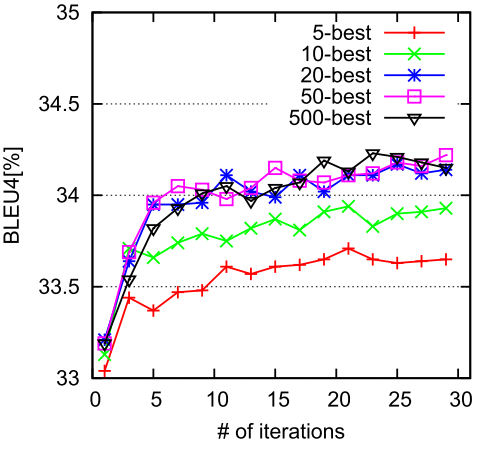

Fig. 23. BLEU curves with different sizes of k-best list (Boosting and phrase-based).

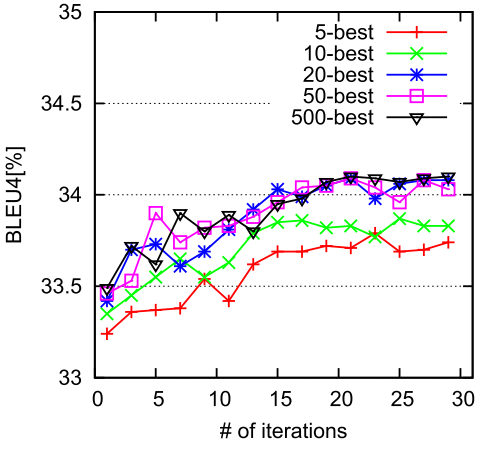

Fig. 24. BLEU scores with different sizes of k-best list (Boosting and Hiero).

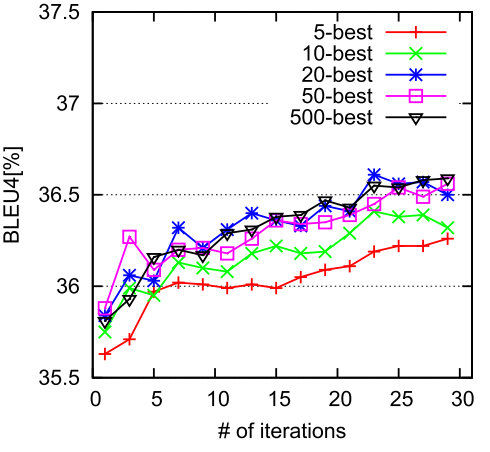

Fig. 25. BLEU scores with different sizes of k-best list (Boosting and syntax-based).

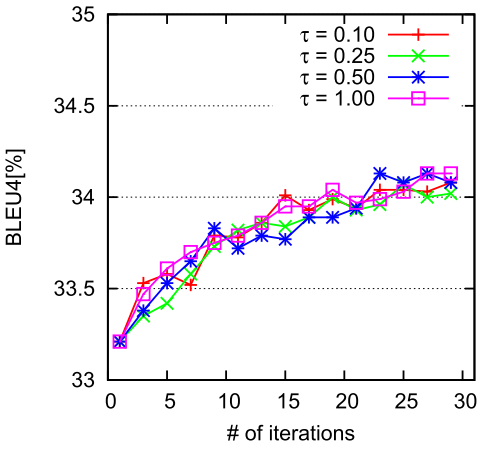

Fig. 26. BLEU curves with different τ (phrase-based).

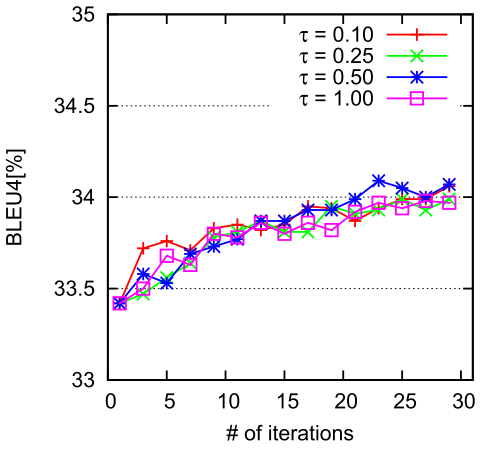

Fig. 27. BLEU curves with different τ (Hiero).

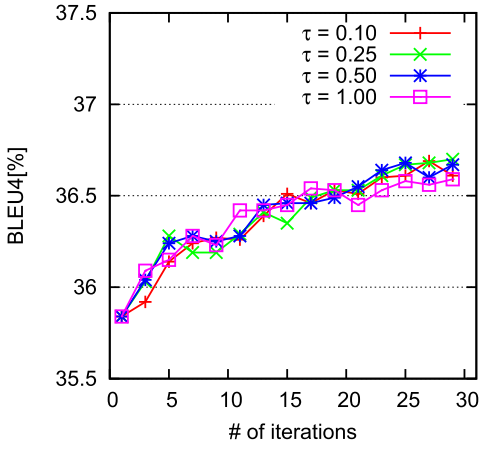

Fig. 28. BLEU curves with different τ (syntax-based).

Bagging-based approach is not very sensitive to the size of the re-sampled training set. In addition, the above observation also suggests that it would be a very nice way to speed up the training by choosing a smaller τ, especially when our approach is applied to practical SMT systems.

We then adjust parameter p (Eq. (10)) to study the impact of the loss function for the Boosting-based approach (Figs. 29–31). In comparison with the default setting (i.e., $p = 5$), a smaller p (i.e., $p = 1$ or 3) leads to degraded BLEU scores. This result suggests that our approach can benefit from more translation candidates used in calculating the losses. However, setting p larger than 10 does not help because of the more noise introduced. Note that when $p = 1$ the loss used in this work is the same as that used in [17]. Here we extend Chiang et al.'s definition [17], and empirically demonstrate that the loss can be better estimated by taking more translation candidates into account.

4.10. Combining multiple translation engines

Although our approach is originally designed to improve individual translation engines, it is also applicable to the combination of multiple structurally different engines. So, in this set of experiments, we investigate the effectiveness of our approach when it is employed to combine multiple SMT engines.

See Table 6 for the evaluation results when the three types of SMT engines are combined together. For comparison, BLEU scores of the three single engines are also reported. In Table 6, the entries labeled with *combined* stands for the combination of various baseline systems using the default method of strong system generation, *Bagging* or *Boosting* stand for the combination of the member systems generated by the Bagging-based approach or the Boosting-based approach, and *All* stands for the combination of all the member systems generated by the two approaches.

As can be seen from the table, using structurally different systems is very beneficial to SMT system combination. When we combine the outputs of the baseline systems (row 4), there are +0.9–3.5/+0.5–2.6/+1.7–4.1/+1.0–2.8 BLEU improvements on the four data sets, respectively. When confusion networks are used (rows 5 and 6), larger improvements are obtained due to the more sophisticated combination models.

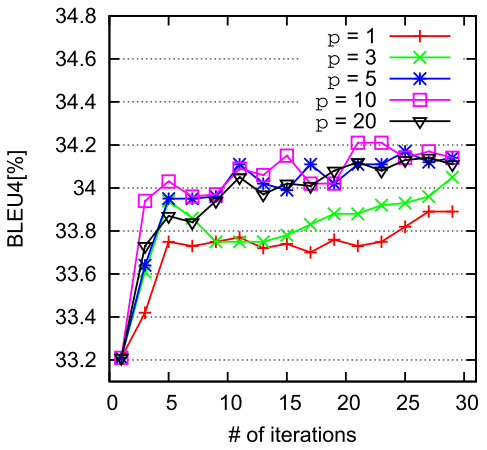

Fig. 29. BLEU curves with different p (phrase-based).

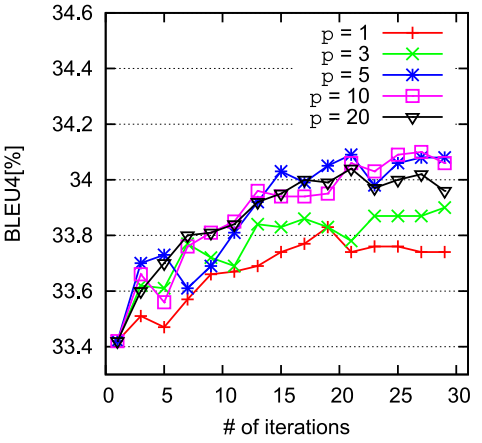

Fig. 30. BLEU curves with different p (Hiero).

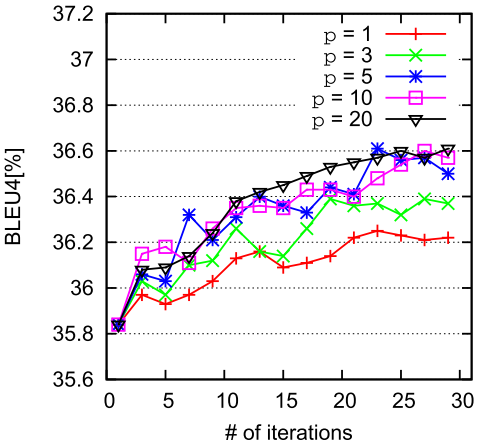

Fig. 31. BLEU curves with different p (syntax-based).

Table 6
Results of combining multiple structurally-different SMT engines. + or ++ = significantly better than all of the single/combined baseline systems (the first four rows) or the confusion network-based counterparts (significance level is 0.05).

		BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Baseline	Phrase-based (single system)	33.21	33.68	32.68	30.59
	Hiero (single system)	33.42	34.30	33.24	30.62
	Syntax-based (single system)	35.84	35.71	35.11	32.43
	Combined (default, 3 member systems)	36.78	36.26	36.85	33.41
Confusion network	IHHM (3 member systems)	37.21	36.83+	36.95	34.01+
	METEOR (3 member systems)	36.95	36.75+	36.99	33.78+
Our approach	Bagging (3 × 30 member systems)	37.31+	37.02+	37.35+	34.13++
	Boosting (3 × 30 member systems)	37.40+	37.18++	37.26+	34.07+
	All (3 × 60 member systems)	37.88++	37.51++	37.65+	34.51++

As expected, the biggest improvements appear when all the member systems are employed for combination. Our approach obtains improvements of 1.1/1.3/0.7/0.9 BLEU points over all four of the baselines on the four data sets respectively, even outperforms the confusion network-based counterparts in all cases. Compared with *Baseline* of the single translation engines, it finally achieves +2.0–4.6/+1.8–3.9/+2.5–4.9/+2.0–3.9 absolute BLEU improvements. These results reflect again the fact that the large number of diverse outputs is a crucial factor to system combination, no matter these outputs are generated from a single engine or from multiple structurally different engines. In some cases, a simple combination system that uses many diverse member systems could even outperform the sophisticated ones that use fewer member systems.

Table 7
Comparison of various methods to select/generate final translation for the phrase-based system. Boldface means the best result.

Member system generation	Strong system generation	BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Bagging	Default	34.11	34.82	33.67	31.55
	MBR	34.06	34.80	33.88	31.53
	Confusion network (IHHM)	**34.43**	**35.21**	**33.99**	**31.80**
	Confusion network (METEOR)	34.10	34.99	33.78	31.52
	Boosted Re-ranking	33.53	34.23	33.01	30.85
	Discriminative Re-ranking	33.66	34.46	32.94	30.94
Boosting	Default	34.12	35.16	33.76	31.59
	MBR	34.19	35.34	33.65	31.81
	Confusion network (IHHM)	**34.34**	**35.39**	**34.12**	**31.93**
	Confusion network (METEOR)	34.00	35.30	33.98	31.73
	Boosted Re-ranking	33.67	34.56	33.22	30.80
	Discriminative Re-ranking	33.58	34.87	33.33	30.71

Table 8
Comparison of various methods to select/generate final translation for the Hiero system. Boldface means the best result.

Member system generation	Strong system generation	BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Bagging	Default	33.98	34.93	33.96	31.36
	MBR	33.90	35.02	33.86	31.36
	Confusion network (IHHM)	**34.24**	**35.45**	**33.99**	**31.65**
	Confusion network (METEOR)	34.10	34.90	33.85	31.48
	Boosted Re-ranking	33.42	34.53	33.49	30.89
	Discriminative Re-ranking	33.48	34.23	33.52	30.67
Boosting	Default	34.05	34.99	34.05	31.30
	MBR	33.97	35.21	33.95	31.09
	Confusion network (IHHM)	**34.36**	**35.44**	34.20	**31.52**
	Confusion network (METEOR)	34.00	35.19	**34.26**	31.13
	Boosted Re-ranking	33.43	34.69	33.47	30.77
	Discriminative Re-ranking	33.51	34.74	33.23	30.50

4.11. Effect of final system generation

So far, we have shown that generating diversified member systems is a key factor to system combination and have demonstrated the effectiveness of Boosting and Bagging in doing this. A natural next question is: given a set of member systems, how does the final system generation impact the translation quality. In the previous experiments, only a simple sentence-level combination method is used in strong system generation of our Bagging/Boosting-based combination framework. As stated in Section 3.4, all the other system combination and re-ranking methods are also reasonable solutions to this issue. It would be interesting to compare different methods to generate the final system. Fortunately, most of the methods described in Section 4.2 are directly applicable to the final system generation in our case.[24] The only exception is MBR which is originally designed to minimize the loss of the output for a single translation system. To adapt MBR to the re-ranking of k-best outputs of multiple systems, we use the multi-system minimum Bayes-risk (or MBR system combination) approach proposed in [42]. This approach uses a linear combination of member systems' probability distributions, and thus allows us to search for the minimum risk translation among the k-best outputs of multiple translation systems.

We then run those re-ranking and system combination systems on the same set of translation candidates generated using Bagging or Boosting. The results for different types of SMT systems are shown in Tables 7–10. We see, first of all, that confusion network-based methods achieve the best BLEU result in most cases. This is because confusion network can produce new translations differing from any of the original translations in the candidate set. Surprisingly, our default method and the MBR method obtain comparable BLEU scores with the confusion network-based counterparts, even outperform some of them (e.g., the system using METEOR) in a few cases. This observation reflects a fact that even a simple combination strategy is still effective in combining member systems generated using Bagging and Boosting. More interestingly, it is observed that both *Discriminative Re-ranking* and *Boosted Re-ranking* are not so effective on this task. The relatively lower BLEU score of traditional re-ranking methods may be due to the lack of the use of consensus information which is generally very useful in combining outputs of multiple SMT systems.

[24] It should be noted that the boosted re-ranking and discriminative re-ranking methods are designed to rank the k-best candidates for the same translation model. They are not applicable to the case where multiple structurally different SMT systems are involved due to their different feature spaces.

Table 9
Comparison of various methods to select/generate final translation for the syntax-based system. Boldface means the best result.

Member system generation	Strong system generation	BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Bagging	Our approach	36.57	36.65	35.88	33.27
	MBR	36.45	36.76	35.75	33.44
	Confusion network (IHHM)	**36.63**	36.99	**36.12**	33.27
	Confusion network (METEOR)	36.52	**37.03**	35.97	**33.45**
	Boosted Re-ranking	36.07	36.04	35.38	32.73
	Discriminative Re-ranking	35.93	36.24	35.61	32.53
Boosting	Our approach	36.52	36.81	35.71	33.46
	MBR	36.45	36.84	**35.99**	33.47
	Confusion network (IHHM)	36.64	**36.93**	**35.99**	**33.64**
	Confusion network (METEOR)	**36.65**	36.74	35.95	33.56
	Boosted Re-ranking	36.22	36.09	35.20	33.06
	Discriminative Re-ranking	36.00	36.04	35.03	32.71

Table 10
Comparison of various methods to select/generate final translation for the combination of the three SMT systems. Boldface means the best result.

Member system generation	Strong system generation	BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Bagging	Default	37.31	37.02	37.35	34.13
	MBR	37.24	37.19	37.29	33.98
	Confusion network (IHHM)	37.40	**37.27**	**37.67**	**34.35**
	Confusion network (METEOR)	**37.45**	37.09	37.39	34.01
	Boosted Re-ranking	N/A	N/A	N/A	N/A
	Discriminative Re-ranking	N/A	N/A	N/A	N/A
Boosting	Default	37.40	37.18	37.26	34.07
	MBR	37.49	37.10	37.41	**34.17**
	Confusion network (IHHM)	37.46	**37.52**	**37.47**	34.13
	Confusion network (METEOR)	**37.52**	37.24	37.20	34.21
	Boosted Re-ranking	N/A	N/A	N/A	N/A
	Discriminative Re-ranking	N/A	N/A	N/A	N/A
All	Default	37.88	37.51	37.65	34.51
	MBR	37.98	37.76	37.53	34.56
	Confusion network (IHHM)	**38.01**	**37.84**	**37.72**	**34.69**
	Confusion network (METEOR)	37.91	37.50	37.71	34.62
	Boosted Re-ranking	N/A	N/A	N/A	N/A
	Discriminative Re-ranking	N/A	N/A	N/A	N/A

4.12. Scaling to larger data-sets

To test the effect of our approach on larger (bilingual) corpora, we conduct experiments on another data configuration. This configuration uses the NIST portion of the bilingual data available for the NIST 2008 track translation task,[25] which consists of about 1.8 million sentence pairs. Other settings are the same as those used in the previous experiments.

Table 11 shows that both Bagging and Boosting are still effective when we switch to the large bilingual corpus. On improving single SMT engine, significant improvements are observed in most cases. Those improvements persist when the three types of SMT systems are involved in combination. On the other hand, our combination system achieves +2.6–3.6/+2.5–3.0/+2.5–3.8/+2.2–3.7 BLEU improvements over the "single-engine" baselines on the four evaluation sets, respectively. Finally we replace the default method of strong system generation with the state-of-the-art confusion network-based methods. The resulting systems (last two rows) achieve the best BLEU scores among all the competing systems, even significantly outperforms all the single and combination baselines in most cases.

5. Related work

5.1. Bagging and Boosting for machine translation

Generally speaking, Bagging and Boosting are two popular instances of the framework of ensemble learning [10,35,37,52,82]. Due to their theoretical performance guarantees and strong experimental results, they have been widely used in various machine learning tasks [1,5,7,22,100], and have been successfully adopted in several NLP applications, such as part-of-speech

[25] LDC catalog number: LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92.

Table 11
Results of scaling to large bilingual corpus. + or ++ = significantly better than the first five baselines or all of the baselines (significance level is 0.05). Boldface means the best result in each set of experiments.

		BLEU4[%]			
		MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Phrase-based (single engine)	*Baseline*	40.51	39.34	38.09	34.45
	Baseline + Fulldev	40.20	39.61	38.17	34.70
	Baseline + 600best	40.67	39.50	38.25	34.71
	Baseline + Batch	40.72	39.62	38.54	34.65
	Baseline + Timefirst	40.51	39.49	38.07	34.59
	MBR	40.75	39.55	38.62	34.64
	Boosted Re-ranking	40.55	39.24	38.24	34.55
	Discriminative Re-ranking	40.49	39.47	38.20	34.55
	Bagging	41.40++	40.67++	**39.15**++	35.39++
	Boosting	**41.63**++	**40.79**++	39.08+	**35.61**++
Hiero (single engine)	*Baseline*	40.89	39.57	38.56	34.77
	Baseline + Fulldev	40.56	39.78	38.81	34.89
	Baseline + 600best	40.98	39.77	38.70	34.92
	Baseline + Batch	41.05	39.93	38.80	34.91
	Baseline + Timefirst	41.01	39.62	38.77	34.71
	MBR	41.04	39.89	38.85	34.93
	Boosted Re-ranking	40.80	39.63	38.41	34.85
	Discriminative Re-ranking	40.90	39.50	38.55	34.70
	Bagging	41.66++	**40.39**++	39.49++	35.47++
	Boosting	**41.74**++	40.35	**39.65**++	**35.70**++
Syntax-based (single engine)	*Baseline*	41.54	39.92	39.36	35.79
	Baseline + Fulldev	41.50	40.09	39.66	35.94
	Baseline + 600best	41.70	40.11	39.54	35.88
	Baseline + Batch	41.74	40.19	39.61	35.97
	Baseline + Timefirst	41.55	40.03	39.44	35.81
	MBR	41.75	40.24	39.71	35.93
	Boosted Re-ranking	41.45	39.93	39.49	35.70
	Discriminative Re-ranking	41.56	39.89	39.47	35.97
	Bagging	42.33++	40.88+	39.98	36.67++
	Boosting	**42.51**++	**41.01**++	**40.10**+	**36.71**++
Phrase-based + Hiero + Syntax-based (multi-engine)	*Baseline*	42.85	41.05	40.95	37.01
	Baseline + Fulldev	42.59	41.39	41.11	37.24
	Baseline + 600best	42.97	41.11	41.03	37.17
	Baseline + Batch	42.99	41.41	41.16	37.27
	Baseline + Timefirst	42.87	41.30	40.97	37.09
	Confusion network (IHHM)	43.24	41.64	41.44	37.41
	Confusion network (METEOR)	43.01	41.47	41.24	37.30
	Bagging	43.77+	41.87+	41.64+	37.84+
	Boosting	43.92++	42.03+	41.65+	37.98++
	All (default)	44.15++	42.44++	41.93+	38.03++
	All (confusion network with IHHM)	**44.27**++	**42.47**++	**42.15**++	**38.15**++
	All (confusion network with METEOR)	44.17++	42.32++	41.97+	38.02++

tagging [2], syntactic parsing [48], named entity classification [19] and coreference resolution [104]. However, the situation is not so pleasant when we switch to SMT. For example, the training criterion of SMT system is generally a discontinuous and non-differentiable function, which leads to very different training paradigms from those of classification systems. Also, combining multiple SMT systems is much more complex than voting from classifiers. While Bagging and Boosting have been shown to be very promising in related tasks, it is not trivial to adapt them to SMT.

To our knowledge, only a few previous studies attempt to introduce Bagging and Boosting into SMT. Perhaps the most related work to this article is [30]. They boosted their re-ranking system by repeatedly training a (log-)linear model on the k-best output of a phrase-based SMT system. [30] has much in common with this article. For example, they both employ ensemble learning, such as Boosting, to improving their weak systems. Also, both articles restrict weak learners to (log-)linear models. As a "bonus", using Bagging and Boosting enables non-linear decision boundaries and ranking models, and results in models that have greater learning capacity than standard (log-)linear models. On the other hand, there appear to be obvious differences between our approach and Duh and Kirchhoff's. First, Duh and Kirchhoff restricted themselves to the re-ranking of k-best output (introducing diversity into their re-rankers), while we focus on using diverse log-linear models during decoding (introducing diversity into decoder and MT output). Although Duh and Kirchhoff's approach is much less costly in terms of running time, it suffers more from search errors due to the limited scope of k-best list. Actually, Duh and Kirchhoff did not address the issue of introducing diversity into MT output, but instead directly applied Boosting to the

late-stage re-ranking. As is shown in our experiment, such an approach may limit the capability of Boosting since there are generally few variations and many redundancies in k-best output of MT systems. In comparison with re-ranking, boosting SMT decoders is relatively more helpful in introducing output diversity and thus improving the translation accuracy. Second, Duh and Kirchhoff's approach is only designed for improving single SMT engines, and cannot be applied to the combination of multiple translation engines that do not share the same feature space. In contrast, we develop a more general solution to generating and combining diverse member systems, no matter they are produced by the same translation engine or multiple structurally different translation engines.

In addition to [30], Lagarda and Casacuberta [62] also applied the Boosting algorithm to SMT. In principle their work is motivated by the same idea as those used in [30] and this work. For example, they tried to combine an ensemble of (log-)linear models to form a stronger model. However, [62] used a simple weighted voting strategy to generate the final hypothesis for output, and thus did not benefit from state-of-the-art approaches in system combination. Moreover, their work is lack of studies and discussions on the diversity issue which is crucial to the success of ensemble learning in MT. Another limitation is that they restricted themselves to phrase-based models through their work. Although their method of error calculation can be extended to handle hierarchical phrase-based models, it is not so straightforward to be applied to modern syntax-based models.

As discussed above, Bagging and Boosting, though not new to SMT, are still one of the juiciest topics in SMT. However, past work on applying Bagging or Boosting to SMT only makes a preliminary attempt to address this issue, and fails to build a general connection between ensemble learning and SMT. In comparison with previous work, we propose a more general solution to the adaptation of Bagging and Boosting to SMT, and present an in-depth, quantitative study of this issue in various SMT systems.

5.2. Generating diverse translations using single SMT engine

In this article, one of our arguments is that diversified translation outputs can be generated from a single translation engine using Bagging and Boosting. Actually, there are also some other studies on building diverse translation systems from a single translation engine for system combination. The first attempt was [69]. They empirically showed that diverse translation systems could be generated by changing word alignment results before rule/phrase extraction. Zhang et al. [113] investigated the effects of Chinese word segmentation on Chinese–English translation and alignment tasks, and demonstrated that both SMT and alignment systems could benefit from diverse word segmentation specifications. Duan et al. [28] proposed a method to generate different translation models for model combination. Their study shows that diverse models could be built by simply sampling over the bilingual training data. However, all these approaches require diverse SMT systems to be generated by changing the parameters at early stages in training, which in turn greatly burdens the whole learning procedure of SMT systems. In contrast, our approach only repeats MERT without additional computation costs and can be further improved using several acceleration methods (see Section 6.1).

Another related work is [26] in which a feature sub-space method was proposed to build a group of translation systems from various different sub-models of an existing SMT engine. However, Duan et al.'s method relies on the heuristics used in feature sub-space selection. For example, they used the remove-one-feature strategy and varied the order of n-gram language model to obtain a satisfactory group of diverse systems. Compared to [26], a main advantage of our approach is that it is applicable to most SMT systems without any heuristics designed for specified systems.

5.3. Re-ranking and MBR

In addition to building an ensemble of diverse member systems, our approach also requires a step of selecting or generating the best translation from the translation candidates produced by member systems. For example, our default method of strong system generation uses a simple scoring function that selects the final output from the candidate pool (Section 4.2). This procedure is to some degree similar to re-ranking in multi-pass decoding, where in the first pass a number of translation candidates are generated using a baseline model, and in the second stage additional features or models are used to select the best translation from the candidate set. As re-ranking provides a simple way to introduce various sophisticated features and machine learning methods into machine translation, it has been widely adopted over the last decade. Before [30], many re-ranking methods had been developed. For example, Och et al. [80] used minimum error rate training for re-ranking and investigated the effectiveness of various features in improving their phrase-based SMT system. Other popular approaches employ discriminative training to learn ranking models that distinguish good translations from bad ones [97]. Nguyen et al. [77] explored other machine learning approaches, such as kernel methods, to relax the restriction on features in a log-linear model. Hasan et al. [44] studied an interesting issue of how big k-best list can be useful in re-ranking for SMT systems. Recently, Duh et al. [31] recast re-ranking as a multi-task learning problem and investigated methods to ease discriminative training when a large number of features are involved.

Besides traditional re-ranking methods, another line of research tries to introduce minimum Bayes-risk decoding into k-best list re-ranking [60]. By minimizing the loss (or expected error) of translation, the MBR decoding method seeks the consensus translation from the k-best list of candidates (or translation lattice), and has been shown to give BLEU improvement to several SMT systems [60,61,103,111].

In principle, all of the above methods are applicable to selecting the final translation from the candidate ensemble in our case. However, traditional re-ranking methods do not consider the consensus information (or n-gram agreement) between translation candidates. As shown in our experiments, this limits their ability in improving BLEU score when the candidate set is made up of the outputs of multiple member systems (with different feature weights). Another note is that our default method of strong system generation is doing something rather similar to MBR decoding. A difference with MBR lies in that our default method does not rely on the posterior probability of hypothesis (i.e., all hypotheses are assigned an equal posterior probability), while MBR require such information for calculating the expected error. This allows us to easily handle the case when multiple member systems are involved, and provides an obvious advantage over traditional MBR which is only work for single translation system. Interestingly, we find that some researchers have been aware of the limitation of MBR and tried to extend MBR to system combination in recent studies [27,42]. For example, Duan et al. [27] used a mixture of different SMT models for decoding. Gonzalez-Rubio et al. [42] straightforwardly combined the expected error of each member system for multi-system generalization of the MBR decoding. As shown in our experiments, like traditional MBR, these methods can also be equipped with our proposed Bagging/Boosting-based framework.

5.4. System combination using multiple SMT engines

In machine translation, researchers have been concerned for years about combining multiple translation systems as well as their outputs for generating better translations than does any individual system [6,76,89]. In general, a combination system takes k-best translation outputs as input, and produces a refined translation using some combination strategies. Several methods have been studied. One of them is sentence-level combination (or *hypothesis selection*) which simply selects one of the translations from the original k-best translation outputs [49,98]. These methods, though simple to implement, have been demonstrated to be very effective on several translation tasks, such as Chinese–English and Arabic–English translation. On the other hand, word-level/phrase-level combination could generate new translations that do not appear in any of the input k-best lists. Most state-of-the-art methods in this scenario use *confusion networks* to represent an exponentially large hypothesis set, and choose the best output by decoding over the confusion networks [6,34,46,64,75,76,88,89]. More recently, Li et al. [63] and Liu et al. [67] explored methods that combine translation hypotheses during the decoding process for individual systems instead of combining their k-best outputs. They demonstrated that their systems could outperform the state-of-the-art word-level combination systems. Although various approaches have been studied, most of them focus on using existing individual systems based on different models or implementations. In contrast, we present a systematic study on generating diversified systems from a single engine in a principled way. One more good thing is that traditional system combination methods actually do not compete with this work. Instead, they are all applicable to our proposed framework. For example, in Sections 4.11 and 4.12, we demonstrate that the translation quality can be improved further when confusion networks are adopted in our framework.

6. Discussion

6.1. Optimization

If implemented naively, the translation speed of the final system will be very slow. For an input sentence, each member system has to decode it independently, and the decoding time is proportional to the number of member systems generated by our approach. Fortunately, with the thought of computation, several optimizations can make the system much more efficient in practice.

A simple solution is to run member systems in parallel. As all the member systems share the same data resources, such as n-gram language model and phrase/rule table, we only need to keep one copy of the required resources in memory. Therefore, the translation speed only depends on the computing power of parallel computation environment, such as the number of CPUs. In addition to accelerating the decoding process, parallel processing can also make the training of the Bagging-based approach much faster since Bagging does not require member systems to be generated sequentially.

Furthermore, we can use joint decoding techniques [67] to save the computation cost of equivalent translation hypotheses shared among member systems. In joint decoding, the search space is structured as a translation hypergraph [51,53] where member systems can share their translation hypotheses. If more than one member systems share the same translation hypothesis, we just need to compute the corresponding feature values only once, rather than repeating the computation in individual decoders. In our experiments, we find that over 60% of the translation hypotheses can be shared among member systems when the number of member systems is more than 4. This result indicates that a promising speed improvement can be achieved by using the joint decoding and hypothesis sharing techniques.

Another method to speed up the system is to accelerate n-gram language model with n-gram caching techniques [33,85]. In this method, an n-gram cache is used to store the most frequently and recently accessed n-grams. When a new n-gram is accessed during decoding, the cache is checked first. If the required n-gram hits the cache, the corresponding n-gram probability is returned by the cached copy rather than re-fetching the original data in the language model. As the speed of SMT decoder depends heavily on the computation of n-gram language model, the acceleration of n-gram language model could lead to substantial speed-up of SMT systems. In our implementation, the n-gram caching in general brings us a speed improvement of over 30%.

6.2. Stopping criteria

Another issue is how to decide when to stop the training. The answer is especially important when our approach is applied to practical systems. Although designing an appropriate stopping criterion is an open issue for Bagging and Boosting [95,96], there are several empirical methods to solve it. A straightforward solution is to use the tendency of BLEU improvement on a development set to determine the stopping point. For example, we can conduct the training until the BLEU improvement in recent iterations is below a threshold. Our experimental study shows that the stable and satisfactory BLEU improvements are achieved in 6–8 iterations, while the largest BLEU improvements are achieved after 20 iterations. According to these empirical results, we can choose the number of iterations as needed. Also, the BLEU score of oracle translation can be considered as another stopping criterion, since it implies how much space is left for further improvement.

In addition to BLEU improvement, the number of new translations is also a good indicator for deciding an appropriate number of training iterations. When a new member system is generated, we count the translations that are never seen in the outputs of previous member systems. The number reflects how many diversified translations are newly introduced and thus how much further improvement we can expect. The training can be ended when the number is below an empirically-determined threshold.

6.3. Instability of SMT training

In general, Bagging is more useful for unstable learners. Also, Boosting is able to benefit from unstable base learning methods, though it requires less instability than Bagging [11,12,22,99]. By increasing the stability of unstable learning systems, both Bagging and Boosting can generate more stable systems by using ensemble methods. Particularly for Bagging, reducing variance of unstable learning systems is one of the major reasons why it works [38].

The issue of instability has been well studied and experimented with the classification task [11,12]. For example, Breiman [11] showed that, compared to decision trees and neural networks, Bagging is less effective in improving nearest neighboring classifiers due to its low instability. Along this line, we further discuss this issue in the context of SMT. Our empirical study proves that SMT training is very sensitive to the distribution of training set and consequently unstable.

Actually, the instability of SMT training has been implicitly used to generate diversified SMT systems for system combination in previous studies [26,28,69], though no in-depth discussion on this issue has been presented so far. For example, Macherey and Och [69] successfully used different word alignment results to generate diversified SMT systems; Zhang et al., [113] demonstrated that it is beneficial to combine SMT systems that are trained using different specifications of word segmentation; Duan et al. [26] showed that the feature sub-space method was very effective to increase the diversity among SMT systems; Duan et al. [28] used different sub-sets of bilingual data to obtain diverse SMT models for model combination. All these studies suggest that SMT training is unstable no matter in the early stages of training pipeline or in the stage of weight tuning.

6.4. Bagging and Boosting full SMT systems

As stated in Section 3.1, we restrict instance weighting to the weight tuning step in this work. Several components within a general SMT training pipeline, such as word alignment and phrase/rule extraction, are trained only once, with uniform weights on the bilingual training sentence-pairs. In other words, instead of applying Bagging and Boosting to the whole procedure of SMT training, we only use them in one of the key training steps. This is not quite the case of standard ensemble methods, where all instances are re-weighted when a full system is re-trained.

Fortunately, with a small modification, our approach is able to handle a more general case where Bagging and Boosting are also performed on the bilingual training sentences. In a standard training framework of SMT system [14,58], phrase/rules are obtained from bilingual training corpus using heuristics. The parameters of these phrase/rules (i.e., feature values in the log-linear model) are then estimated with MLE. Once training sentence pairs are weighted, the above phrase/rule extraction and parameter estimation procedure can proceed as usual, but with weight counts. To train our SMT systems with weighted training sentence-pairs, we use the method presented in [74]. The basic idea is that, instead of counting each occurrence of a phrase/rule as unit one in MLE, we use the weight of the corresponding sentence-pair where the phrase/rule is extracted. Motivated by this method, we update the parameter estimation program to support weighted sentence-pairs in training. For the other components of the Bagging/Boosting-based learning system, we use the same setting as those adopted in our previous experiments.

Then we conduct an additional experiment to study the ability of our approach in improving full SMT systems. As we have to decode all the training sentences to obtain the error rate for (Boosting-based) re-weighting, it is very time consuming to run our program on very large corpus. To finish the experiment in an acceptable time, we use a training set consisting of 50,000 sentences (less than 30 words on the source language side) that are randomly selected from the original training corpus, and set the iteration number to 6 for both Bagging and Boosting. Besides, we choose the phrase-based system as the base SMT system since it is much faster than the Hiero and syntax-based systems. But we believe that the result obtained here is a good reference for other types of SMT systems.

Table 12 shows that applying Bagging and Boosting to early-stage training is promising. The biggest BLEU improvement is obtained when our approach is employed in both parameter estimation and weight tuning. Moreover, it is observed

Table 12
BLEU scores of Bagging and Boosting the phrase-based system on the bilingual training corpus and the development set. The BLEU score of training set is calculated on 5000 sentences that are randomly selected from the training data. + or ++ = significantly better than the system that uses baseline methods for parameter estimation and weight training (significance level is 0.05 or 0.01).

Training on bilingual corpus	Weight training (or tuning) on development set	BLEU4[%]				
		Bilingual (Train)	MT03 (Dev)	MT04 (Test)	MT05 (Test)	MT06 (Test)
Baseline	Baseline	42.40	30.02	30.56	29.23	28.12
Baseline	Bagging	42.75	30.73+	31.37+	29.67+	28.50
	Boosting	42.87+	31.14++	31.53++	29.82+	28.66+
Bagging	Baseline	43.22++	30.69++	31.43++	29.70+	28.74+
Boosting		43.81++	30.42	31.11+	29.49	28.47
Bagging	Bagging	43.24++	30.97++	31.59++	29.91+	28.92++
	Boosting	43.20++	31.34++	31.73++	30.05++	28.80+
Boosting	Bagging	43.37++	31.17++	31.69++	29.92+	28.77+
	Boosting	43.30++	31.29++	31.77++	29.89+	28.98++

that Boosting has a lower BLEU score than Bagging on test sets when our approach is used for parameter estimation only. This interesting phenomenon might be due to the noisy training corpus used in this study. We find that there are many inappropriate translations and even incomplete translations in our bilingual data. It is likely to lead to inaccurate estimation of error rate for Boosting, especially when only one reference translation is provided for BLEU calculation. As suggested in [36], this problem may result in the overfitting of training set since later learner over-emphasizes examples that are noise, and thus creates member systems that generalize poorly on test sets.

7. Limitations and future work

Naturally, as with any technique, the proposed approach has certain limitations. Several issues need further investigation. Specifically, the approach presented in this article is lack of theatrical verification, though its idea is intuitively explained and its effectiveness is empirically demonstrated on several evaluation sets. For example, in (binary) classification problem, it has been proved that the standard Boosting algorithm can obtain arbitrarily low training error rate if sufficient data is provided and the algorithm runs in sufficient rounds [37]. On the other hand, errors on test sets do not increase when many classifiers/systems are combined [93]. However, it is not the case when Boosting is applied to SMT. For example, in this study, the training error (in terms of 1 – BLEU) of our approach is far from zero even when the Boosting algorithm runs for tens of rounds. Also, the convergence of our approach is not guaranteed from the theoretical perspective. These issues suggest very interesting further directions, such as a further study on the theoretical properties of Bagging and Boosting SMT systems.

Another interesting issue we would investigate is how to make better use of translation and n-gram posteriors in final system generation. For example, in our default method of strong system generation, we do not distinguish different translations in the k-best list. However, a k-best list of translations is not simply a set of translations. Instead, it is typically viewed as a weighted list in which each translation has a weight (or probability) indicating the "confidence" that the translation model has on it. A reasonable improvement of this method is to use a weighted list for calculating the loss and consensus-based features, instead of uniformly averaging all the hypotheses in the list. Another problem is that both the default method and the MBR method studied in this article calculate various features and n-gram posteriors over a k-best list of translation candidates, which suffers from the limited scope of k-best output of MT systems. Tromble et al. [103] and Kumar et al. [61] have pointed out that MBR systems can benefit more from the calculation of n-gram posterior over a translation lattice than a limited number of translation hypotheses. Therefore it would be interesting to improve our hypothesis selection systems using lattice-based methods. As all hypothesis selection systems are applicable to the strong system generation in the proposed framework, it is expected to obtain a further improvement by incorporating translation lattices into our approach, such as using lattice-based MBR instead of k-best MBR [61,103].

Finally, since boosting SMT systems is very time consuming, we plan to investigate novel approaches that will retain the benefits of Boosting but be more efficient (e.g., approaches that require fewer iterations for convergence). To do this, we will try to preserve the basic idea of Boosting (e.g., focusing on sentences on which MT system has poor translations) while preventing repeatedly calling the SMT decoder to calculate the error on each individual sentence. One possible approach is to sample over the training data and carry out the Boosting-based training on a small sub-set of training data. Also, we can exclude the training samples that have satisfactory BLEU score in our baseline system, and then concentrate only on the remaining samples from the second round of training. Another approach would be to cluster the training samples into several groups and only consider the representative ones in each group (e.g., centroid of cluster) in training.

8. Conclusions

We have proposed a Bagging/Boosting-based approach to address the issue of building a strong translation system using a group of weak translation systems generated from a single SMT engine. The proposed approach can work with any of

current SMT systems and make them stronger almost "for free". Moreover, most of system combination methods can be fit into the proposed framework to generate the final translation system from the ensemble of component systems. We apply our approach to three types of SMT systems, and conduct evaluations on the NIST Chinese–English MT evaluation corpora. The experimental results show that our approach is very useful in improving the translation accuracy of three state-of-the-art SMT systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based system. Also, the proposed approach shows its ability in improving the performance of existing system combination methods. For example, the biggest improvements are obtained when the weak systems are generated using Bagging/Boosting and the strong system is learned using a state-of-the-art system combination method. More interestingly, it is observed that SMT training is unstable, which explains why Bagging and Boosting work well in this study.

Acknowledgements

This work was supported in part by the National Science Foundation of China (Grant Nos.: 61073140, 61272376), the Natural Science Foundation for the Youth of China (Grant No.: 31000468), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No.: 20100042110031) and the Fundamental Research Funds for the Central Universities. The authors would like to thank the anonymous reviewers of ACL 2010 and the AI journal for their pertinent comments, Chunliang Zhang and Shujie Yao for their valuable suggestions for improving this article, and Tianning Li and Rushan Chen for developing parts of the baseline systems.

References

[1] N. Abe, H. Mamitsuka, Query learning strategies using Boosting and Bagging, in: Proceedings International Conference on Machine Learning (ICML), Wisconsin, USA, 1998, pp. 1–9.
[2] S. Abney, R. Schapire, Y. Singer, Boosting applied to tagging and PP attachment, in: Proceedings Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC), Maryland, USA, 1999, pp. 38–45.
[3] E. Alpaydin, Introduction to Machine Learning, Adaptive Computation and Machine Learning, The MIT Press, Cambridge, 2010.
[4] S. Banerjee, A. Lavie, METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, in: Proceedings ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Edinburgh, Scotland, 2005, pp. 65–72.
[5] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 173–180.
[6] S. Bangalore, G. Bordel, G. Riccardi, Computing consensus translation from multiple machine translation systems, in: Proceedings Automatic Speech Recognition and Understanding Workshop (ASRU), Madonna di Campiglio, Italy, 2001, pp. 351–354.
[7] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants, Machine Learning 36 (1/2) (1999) 105–139.
[8] A.L. Berger, S.A. Pietra, V.J. Pietra, A maximum entropy approach to natural language processing, Computational Linguistics 22 (1) (1996) 39–71.
[9] P.J. Bickel, K.A. Doksum, Mathematical Statistics: Basic Ideas and Selected Topics, Holden-Day Inc., Oakland, CA, USA, 1977.
[10] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[11] L. Breiman, Heuristics of instability and stabilization in model selection, Annals of Statistics 24 (1996) 2350–2383.
[12] L. Breiman, Heuristics of instability and stabilization in model selection, Technical report no. 416, Department of Statistics, University of California, Berkeley, CA, 1994.
[13] P.F. Brown, S.A. Pietra, V.J. Pietra, R.L. Mercer, The mathematics of statistical machine translation: Parameter estimation, Computational Linguistics 19 (2) (1993) 263–311.
[14] D. Chiang, A hierarchical phrase-based model for statistical machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, MI, 2005, pp. 263–270.
[15] D. Chiang, Hierarchical phrase-based translation, Computational Linguistics 33 (2) (2007) 201–228.
[16] D. Chiang, S. DeNeefe, Y.S. Chan, H.T. Ng, Decomposability of translation metrics for improved evaluation and efficient algorithms, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, HI, 2008, pp. 610–619.
[17] D. Chiang, Y. Marton, P. Resnik, Online large-margin training of syntactic and structural translation features, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, HI, 2008, pp. 224–233.
[18] J.H. Clark, C. Dyer, A. Lavie, N.A. Smith, Better hypothesis testing for statistical machine translation: Controlling for optimizer instability, in: Proceedings Annual meeting of the Association for Computational Linguistics (ACL), Portland, OR, USA, 2011, pp. 176–181.
[19] M. Collins, Y. Singer, Unsupervised models for named entity classification, in: Proceedings Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC), College Park, MD, USA, 1999, pp. 100–110.
[20] B. Cowan, I. Kučerová, M. Collins, A discriminative model for tree-to-tree translation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Sydney, Australia, 2006, pp. 232–241.
[21] T.G. Dietterich, Machine learning research: Four current directions, AI Magazine 18 (4) (1997) 97–136.
[22] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting and randomization, Machine Learning 40 (2) (2000) 1–19.
[23] Y. Ding, M. Palmer, Machine translation using probabilistic synchronous dependency insertion grammars, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, MI, 2005, pp. 541–548.
[24] G. Doddington, Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, in: Proceedings ARPA Workshop on Human Language Technology, Santa Monica, CA, USA, 2002, pp. 1–8.
[25] H. Drucker, C. Cortes, L.D. Jackel, Y. Lecun, V. Vapnik, Boosting and other machine learning algorithms, in: Proceedings International Conference on Machine Learning (ICML), New Brunswick, NJ, USA, 1994, pp. 53–61.
[26] N. Duan, M. Li, T. Xiao, M. Zhou, The feature subspace method for SMT system combination, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Suntec, Singapore, 2009, pp. 1096–1104.
[27] N. Duan, M. Li, D. Zhang, M. Zhou, Mixture model-based minimum Bayes risk decoding using multiple machine translation systems, in: Proceedings International Conference on Computational Linguistics (COLING), Beijing, China, 2010, pp. 313–321.
[28] N. Duan, H. Sun, M. Zhou, Translation model generalization using probability averaging for machine translation, in: Proceedings International Conference on Computational Linguistics (COLING), Beijing, China, 2010, pp. 304–312.

[29] N. Duffy, D. Helmbold, Potential boosters? in: Proceedings Advances in Neural Information Processing Systems (NIPS), Granada, Spain, 1999, pp. 258–264.
[30] K. Duh, K. Kirchhoff, Beyond log-linear models: Boosted minimum error rate training for *N*-best Re-ranking, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Columbus, OH, 2008, pp. 37–40.
[31] K. Duh, K. Sudoh, H. Tsukada, H. Isozaki, M. Nagata, *N*-best reranking by multitask learning, in: Proceedings the Fifth Workshop on Statistical Machine Translation and METRICS MATR, Uppsala, Sweden, 2010, pp. 375–383.
[32] J. Eisner, Learning non-isomorphic tree mappings for machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan, 2003, pp. 205–208.
[33] M. Federico, M. Cettolo, Efficient handling of n-gram language models for statistical machine translation, in: Proceedings the Second Workshop on Statistical Machine Translation, Prague, Czech Republic, 2007, pp. 88–95.
[34] Y. Feng, Y. Liu, H. Mi, Q. Liu, Y. Lü, Lattice-based system combination for statistical machine translation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Suntec, Singapore, 2009, pp. 1105–1113.
[35] Y. Freund, Boosting a weak learning algorithm by majority, Information and Computation 121 (2) (1995) 256–285.
[36] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings International Conference on Machine Learning (ICML), Bari, Italy, 1996, pp. 148–156.
[37] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to Boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[38] J.H. Friedman, On bias, variance, 0/1-loss, and the curse-of-dimensionality, Data Mining and Knowledge Discovery 1 (1997) 55–77.
[39] M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, I. Thayer, Scalable inferences and training of context-rich syntax translation models, in: Proceedings Annual meeting of the Association for Computational Linguistics (ACL), Sydney, Australia, 2006, pp. 961–968.
[40] M. Galley, C.D. Manning, A simple and effective hierarchical phrase reordering model, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Hawaii, USA, 2008, pp. 848–856.
[41] V. Goel, W. Byrne, Minimum Bayes-risk automatic speech recognition, Computer Speech and Language 14 (2) (2000) 115–135.
[42] J. Gonzalez-Rubio, A. Juan, F. Casacuberta, Minimum Bayes-risk system combination, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Portland, OR, USA, 2011, pp. 1268–1277.
[43] L. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (10) (1990) 993–1001.
[44] S. Hasan, R. Zens, H. Ney, Are very large *n*-best lists useful for SMT? in: Proceedings Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Rochester, USA, 2007, pp. 57–60.
[45] S. Hashem, Optimal linear combinations of neural networks, Neural Networks 10 (4) (1997) 599–614.
[46] X. He, M. Yang, J. Gao, P. Nguyen, R. Moore, Indirect HMM based hypothesis alignment for combining outputs from machine translation systems, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, HI, 2008, pp. 98–107.
[47] K. Heafield, A. Lavie, Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme, The Prague Bulletin of Mathematical Linguistics 93 (2010) 27–36.
[48] J.C. Henderson, E. Brill, Bagging and Boosting a Treebank Parser, in: Proceedings North American Chapter of the Association for Computational Linguistics (NAACL), Seattle, WA, USA, 2000, pp. 34–41.
[49] A.S. Hildebrand, S. Vogel, Combination of machine translation systems via hypothesis selection from combined *n*-best lists, in: Proceedings Conference of the Association for Machine Translation in the Americas (AMTA), Waikiki, HI, 2008, pp. 254–261.
[50] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[51] L. Huang, D. Chiang, Forest rescoring: Faster decoding with integrated language models, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Prague, Czech Republic, 2007, pp. 144–151.
[52] A.K. Jain, R.P. Duin, J. Mao, Statistical pattern recognition: A review, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1) (2000) 4–37.
[53] D. Klein, C. Manning, Parsing and hypergraphs, in: Proceedings the Seventh International Workshop on Parsing Technologies (IWPT), Beijing, China, 2001, pp. 351–372.
[54] K. Knight, Decoding complexity in word-replacement translation models, Computational Linguistics 25 (4) (1999) 607–615.
[55] P. Koehn, Statistical significance tests for machine translation evaluation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain, 2004, pp. 388–395.
[56] P. Koehn, Statistical Machine Translation, Cambridge University Press, UK, 2010.
[57] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst Moses, Open source toolkit for statistical machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL) Companion Volume Proceedings of the Demo and Poster Sessions, Prague, Czech Republic, 2007, pp. 177–180.
[58] P. Koehn, F. Och, D. Marcu, Statistical phrase-based translation, in: Proceedings Human Language Technology and Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), Edmonton, USA, 2003, pp. 48–54.
[59] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: Proceedings Advances in Neural Information Processing Systems (NIPS), The MIT Press, USA, 1994, pp. 231–238.
[60] S. Kumar, W. Byrne, Minimum Bayes-risk decoding for statistical machine translation, in: Proceedings Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL), Boston, MA, USA, 2004, pp. 169–176.
[61] S. Kumar, W. Macherey, C. Dyer, F. Och, Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices, in: Proceedings Annual Meeting of the Association for Computational Linguistics and Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP), Suntec, Singapore, 2009, pp. 163–171.
[62] A. Lagarda, F. Casacuberta, Applying Boosting to statistical machine translation, in: Proceedings Annual Meeting of European Association for Machine Translation (EAMT), Hamburg, Germany, 2008, pp. 88–96.
[63] M. Li, N. Duan, D. Zhang, C. Li, M. Zhou, Collaborative decoding: Partial hypothesis re-ranking using translation consensus between decoders, in: Proceedings Annual Meeting of the Association for Computational Linguistics and Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP), Suntec, Singapore, 2009, pp. 585–592.
[64] C. Li, X. He, Y. Liu, N. Xi, Incremental HMM alignment for MT system combination, in: Proceedings Annual Meeting of the Association for Computational Linguistics and Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP), Suntec, Singapore, 2009, pp. 949–957.
[65] P. Liang, A. Bouchard-Côté, D. Klein, B. Taskar, An end-to-end discriminative approach to machine translation, in: Proceedings International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING/ACL), Sydney, Australia, 2006, pp. 104–111.
[66] Y. Liu, Q. Liu, S. Lin, Tree-to-string alignment template for statistical machine translation, in: Proceedings International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING/ACL), Sydney, Australia, 2006, pp. 609–616.

[67] Y. Liu, H. Mi, Y. Feng, Q. Liu, Joint decoding with multiple translation models, in: Proceedings Annual Meeting of the Association for Computational Linguistics and Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP), Suntec, Singapore, 2009, pp. 576–584.
[68] A. Lopez, Statistical machine translation, ACM Computing Surveys 40 (3) (2008) 1–49.
[69] W. Macherey, F. Och, An empirical study on computing consensus translations from multiple machine translation systems, in: Proceedings Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL), Prague, Czech Republic, 2007, pp. 986–995.
[70] D. Marcu, W. Wang, A. Echihabi, K. Knight, SPMT: Statistical machine translation with syntactified target language phrases, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Sydney, Australia, 2006, pp. 44–52.
[71] S. Marsland, Machine Learning: An Algorithmic Perspective, Chapman & Hall/CRC Machine Learning & Pattern Recognition, CRC Press, Boca Raton, 2009.
[72] L. Mason, J. Baxter, P. Bartlett, M. Frean, Functional Gradient Techniques for Combining Hypotheses, Advances in Large Margin Classifiers, The MIT Press, USA, 1999.
[73] L. Mason, J. Baxter, P. Bartlett, M. Frean, Boosting algorithms as gradient descent, in: Proceedings Advances in Neural Information Processing Systems (NIPS), Denver, CO, USA, 2000, pp. 512–518.
[74] S. Matsoukas, A.I. Rosti, B. Zhang, Discriminative corpus weight estimation for machine translation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Suntec, Singapore, 2009, pp. 708–717.
[75] E. Matusov, G. Leusch, R.E. Banchs, N. Bertoldi, D. Déchelotte, M. Federico, M. Kolss, Y. Lee, J.B. Mariño, M. Paulik, S. Roukos, H. Schwenk, H. Ney, System combination for machine translation of spoken and written language, IEEE Transactions on Audio Speech and Language Processing 16 (7) (2008) 1222–1237.
[76] E. Matusov, N. Ueffing, H. Ney, Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment, in: Proceedings Annual Conference of the European Chapter of the Association for Computational Linguistics (EACL), Trento, Italy, 2006, pp. 33–40.
[77] P. Nguyen, M. Mahajan, X. He, Training non-parametric features for statistical machine translation, in: Proceedings the Second Workshop on Statistical Machine Translation, Prague, Czech Republic, 2007, pp. 72–79.
[78] S. Nießen, F. Och, G. Leusch, H. Ney, An evaluation tool for machine translation: Fast evaluation for machine translation research, in: Proceedings the Second International Conference on Language Resources and Evaluation (LREC), Athens, Greece, 2000, pp. 39–45.
[79] F. Och, Minimum error rate training in statistical machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan, 2003, pp. 160–167.
[80] F. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, D. Radev, A smorgasbord of features for statistical machine translation, in: Proceedings Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL), New York City, USA, 2004, pp. 161–168.
[81] F. Och, H. Ney, Discriminative training and maximum entropy models for statistical machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, USA, 2002, pp. 295–302.
[82] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of Artificial Intelligence Research 11 (1999) 16–98.
[83] D. Opitz, J. Shavlik, Actively searching for an effective neural-network ensemble, Connection Science 8 (3/4) (1996) 337–353.
[84] K. Papineni, S. Roukos, T. Ward, W. Zhu, BLEU: A method for automatic evaluation of machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, USA, 2002, pp. 311–318.
[85] A. Pauls, D. Klein, Faster and smaller n-gram language models, in: Proceedings Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT), Portland, OR, USA, 2011, pp. 258–267.
[86] A. Ratnaparkhi, Maximum entropy models for natural language ambiguity resolution, Ph.D. thesis, University of Pennsylvania, PA, 1998.
[87] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (2010) 1–39.
[88] A. Rosti, S. Matsoukas, R. Schwartz, Improved word-level system combination for machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Prague, Czech Republic, 2007, pp. 312–319.
[89] A. Rosti, B. Xiang, S. Matsoukas, R. Schwartz, N. Ayan, B.J. Dorr, Combining outputs from multiple machine translation systems, in: Proceedings Human Language Technology and Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), Rochester, New York, 2007, pp. 228–235.
[90] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, NJ, 1995.
[91] R.E. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227.
[92] R.E. Schapire, The Boosting approach to machine learning: An overview, in: D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick, B. Yu (Eds.), Nonlinear Estimation and Classification, Springer, New York, 2003.
[93] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, The Annals of Statistics 26 (5) (1998) 1651–1686.
[94] R.E. Schapire, Y. Singer, BoosTexter: A boosting-based system for text categorization, Machine Learning 39 (2/3) (2000) 135–168.
[95] M. Sebban, R. Nock, S. Lallich, Boosting neighborhood-based classifiers, in: Proceedings International Conference on Machine Learning (ICML), Williamstown, MA, USA, 2001, pp. 505–512.
[96] M. Sebban, R. Nock, S. Lallich, Stopping criterion for Boosting-based data reduction techniques: From binary to multiclass problems, Journal of Machine Learning Research 3 (2002) 863–885.
[97] L. Shen, A. Sarkar, F. Och, Discriminative reranking for machine translation, in: Proceedings Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL), Boston, MA, USA, 2004, pp. 177–184.
[98] K.C. Sim, W. Byrne, M. Gales, H. Sahbi, P. Woodland, Consensus network decoding for statistical machine translation system combination, in: Proceedings International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, HI, 2007, pp. 105–108.
[99] M. Skurichina, Stabilizing weak classifiers: Regularization and combining techniques in discriminant analysis, Ph.D. thesis, Delft University of Technology, Delft, 2001.
[100] M. Skurichina, R. Duin Bagging, Boosting and the random subspace method for linear classifiers, Pattern Analysis and Applications 5 (2) (2002) 1211–1235.
[101] N.A. Smith, Novel estimation methods for unsupervised discovery of latent structure in natural language text, Ph.D. thesis, Johns Hopkins University, Baltimore, MD, 2006.
[102] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul, A study of translation edit rate with targeted human annotation, in: Proceedings Conference of the Association for Machine Translation in the Americas (AMTA), Cambridge, MA, USA, 2006, pp. 223–231.
[103] R.W. Tromble, S. Kumar, F. Och, W. Macherey, Lattice minimum Bayes-risk decoding for statistical machine translation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, HI, 2008, pp. 620–629.
[104] S. Vemulapalli, X. Luo, J.F. Pitrelli, I. Zitouni, Using Bagging and Boosting techniques for improving coreference resolution, Informatica 34 (2010) 111–118.
[105] H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, San Francisco, CA, 2005.

[106] D.H. Wolpert, Stacked generalization, Neural Networks 5 (2) (1992) 241–259.
[107] T. Xiao, J, Zhu, H. Zhang, Q. Li, NiuTrans: An open source toolkit for phrase-based and syntax-based machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL) System Demonstrations, Jeju Island, Korea, 2012, pp. 19–24.
[108] T. Xiao, M. Li, D. Zhang, J. Zhu, M. Zhou, Better synchronous binarization for machine translation, in: Proceedings Conference on Empirical Methods in Natural Language Processing (EMNLP), Suntec, Singapore, 2009, pp. 362–370.
[109] T. Xiao, J. Zhu, M. Zhu, H. Wang, Boosting-based System combination for machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Uppsala, Sweden, 2010, pp. 739–748.
[110] D. Xiong, Q. Liu, S. Lin, Maximum entropy based phrase reordering model for statistical machine translation, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Sydney, Australia, 2006, pp. 521–528.
[111] H. Zhang, D. Gildea, Efficient multi-pass decoding for synchronous context free grammars, in: Proceedings Annual Meeting of the Association for Computational Linguistics (ACL), Columbus, OH, 2008, pp. 209–217.
[112] H. Zhang, L. Huang, D. Gildea, K. Knight, Synchronous binarization for machine translation, in: Proceedings Human Language Technology and Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), New York City, USA, 2006, pp. 256–263.
[113] R. Zhang, K. Yasuda, E. Sumita, Improved statistical machine translation by multiple Chinese word segmentation, in: Proceedings the Third Workshop on Statistical Machine Translation, Columbus, OH, 2008, pp. 216–223.